ADOBE® CONTENT SERVER 4

User Manual

Adobe Systems Incorporated

Version 1.0
11 September 2008

Not for Distribution

Adobe® Content Server User Guide

Contents

Introduction

Adobe Content Server 4 is the latest Adobe server solution for digital rights management (DRM) for eBooks and other digital publications. Content Server 4 provides a robust set of services that can be integrated into a customer's existing infrastructure, including:

- Capability to package unencrypted publications for delivery to customers
- Support for publications in both Adobe® PDF and EPUB formats
- Services to enable integration with distribution services such as stores and libraries

Adobe Content Server 4 is the successor to Adobe Content Server 3 (latest version 3.01), which provided similar services for PDF publications. Based on prevalent usage of Content Server 3, version 4 has been designed and implemented as relatively low-level services that facilitate integration with existing content management systems (CMS) and other infrastructure that the customer has already built up to support their business.

The rest of this manual covers

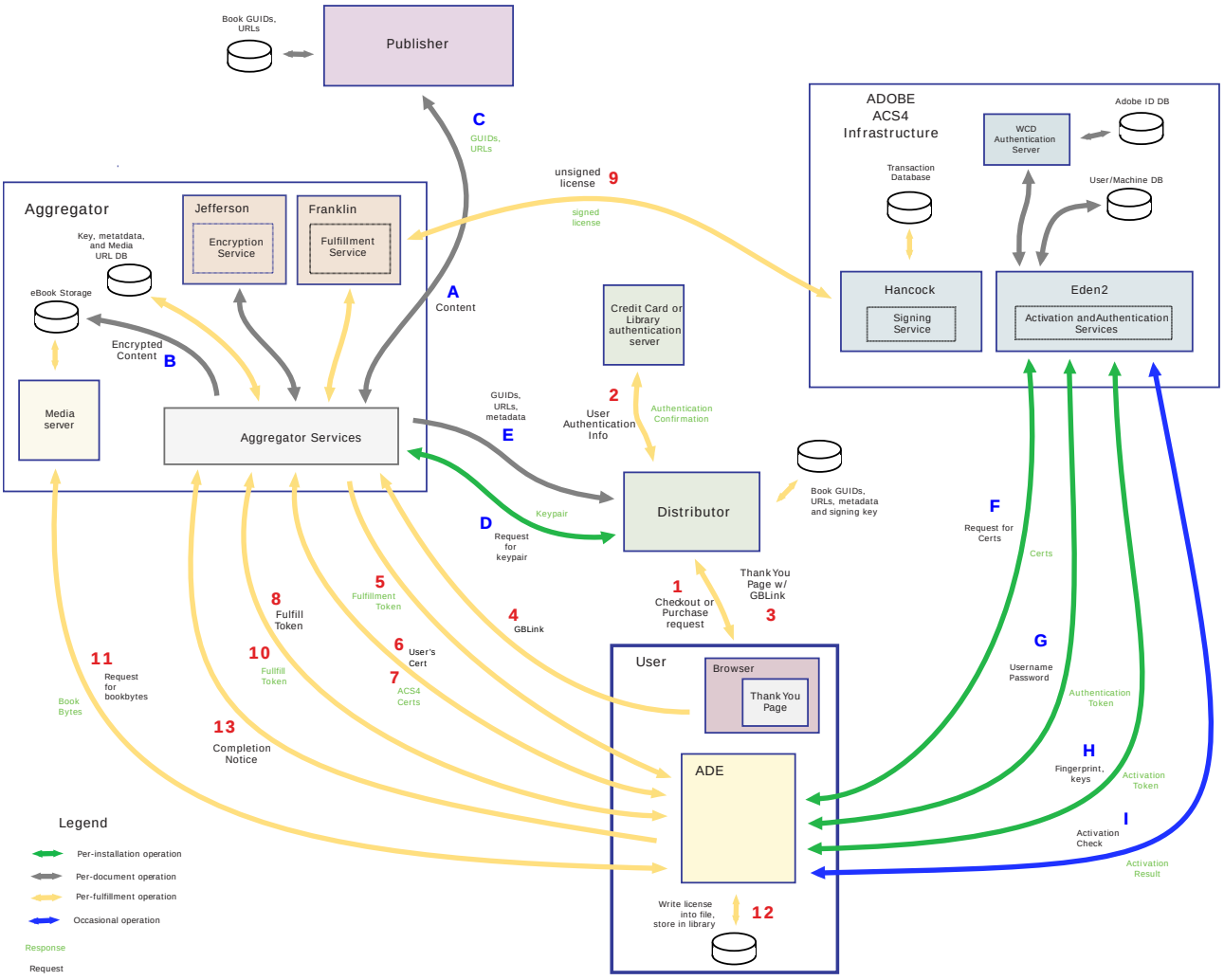
- A brief overview of a typical system
- The system requirements for operation
- The set of deliverables that comprise the system
- A guide to the installation of Content Server 4 on a typical system
- Migrating content from an existing Content Server 3 system
- Packaging new content using Content Server 4
- Digital rights management capabilities of Content Server 4
- The Admin console
- Technical specifications are covered in the Content Server 4 Technical Reference Manual

1 Overview

The primary elements of a complete Content Server 4 system comprise:

- A Publisher, who generates the original (unencrypted) documents.
- An Adobe-hosted server (called Eden2) which provides the two services: activation and authentication
- An Adobe-hosted server (called Hancock) which performs the license-signing service.
- An Aggregator, who hosts a standalone Content Server 4 server. The Aggregator performs the encryption of the ebooks themselves and performs the fulfillments to the users.
- A media server, which is typically a HTTP server that hosts the bookbytes. While the media server is usually operated by the aggregator, the encrypted bookbytes could in principle be hosted elsewhere, distributed via a CDN, but this would be unusual.
- A Distributor (eCommerce or library site) that actually handles the users' requests (e.g. purchase or check-out)

There are many configurations possible, but this is the most common and will be the central theme in this document.



The diagram above (which is also present as vector-based PDF as an attachment to this document) outlines the workflow at a high level. Note that the golden arrows are those that are exercised on a per-transaction basis, while the green arrows those that take place once (e.g. upon installation of ADE or commencement of communication between the publisher and the aggregator), and the grey arrows are those that are only exercised once per document. The one blue arrow is for the occasional authentication check that ADE performs. Also, for flows that are two-way (e.g. HTTP request/response), the request is in black while the response is in green. Naturally, the true diagram of this would be much more complex as it would also include pathways for error handling, etc., but it does illustrate the basic flows for a simple, most-common usage Content Server 4-based system. In addition, there are many possible topologies possible, e.g. the publisher could be operating the Content Server 4 site themselves rather than using an aggregator, but the diagram and associated discussion are based on the most common scenario where the publisher, the aggregator and distributor are all separate entities.

Note that the flows are either numbered or lettered. The numbered flows are those that are per-transaction, while the lettered flows are typically only performed once, e.g. once per-document or once per installation of the client software. The details of the per-document steps are listed in the following table. The details of the numbered steps are explained in the subsequent table.

The following table covers the details of the per-document flow, i.e. those steps that are done only once per document (or, in some cases only once for each publisher or distributor. The workflow described in the table is restricted (for simplicity's sake) to a single book, but most if not all workflows should support batch operations.

Step	Action	Comment
A	A publisher sends unencrypted PDF or EPUB to the Aggregator. The Aggregator then encrypts (packages) the document. This process also includes generating a GUID which uniquely identifies the document. The Aggregator stores the GUID, metadata, decryption key, and URL in a manner in which it can later be retrieved to create a license request in response to a fulfillment transaction request (typically as fields within a per-publication database)	
B	The document is then transferred to a "media server" operated by the Aggregator. The result of this step is a URL to the document's location on the media-server. The GUID, metadata, decryption key and URL are all stored in the Aggregator's database for use during the fulfillment step. The media server, which is usually operated by the Aggregator, is typically a simple HTTP server.	The actual posting/hosting is usually done by the Aggregator, but may instead return the book-bytes to the publisher to be hosted elsewhere.
C	The URL to the encrypted book-bytes is returned to the Publisher, along with the GUID.	This (optional step) is primarily for accounting and QE purposes.
D	The Aggregator's assigns the Distributor's id, the base URL used for fulfillment, and shared secret (used to sign requests). Note that the above is a typical workflow, but in some cases the Aggregator themselves hosts the Distributor site, in which case this is all internal to the Aggregator.	This is done only once per Distributor, not per document
E	The ebook GUID, permissions and the metadata associated with the book are sent to the Distributor. The Distributor stores these away in its own database, from which it can dynamically generate the sales and thank you pages.	
F	User installs ADE 1.6, which automatically calls to the Eden2 Activation server to exchange the keys (certificates) they need to talk with each other securely	
G	The user is then prompted to enter their Adobe ID. When the user authorizes their computer with their Adobe ID and password, the ID, the username, password and activation token are sent to the authentication service on Eden2. The server checks if the username and password already exist in its DB. If so, it responds with those keys to the user. The keys that were generated during the activation step (F) are then discarded. If the username is not in the database, then the service stores the supplied keys in its database.	
H	ADE then calls to the activation server, passing the certificate and the user's machine fingerprint and getting back the activation token for ADE	

I	Occasionally (typically 1% of the time) when the user starts ADE, it makes a call to the authentication server, passing the authentication token, OS and version and locale. The server responds successfully if the authentication is found to be valid in its database. If the authentication is revoked, the server responds with "invalid activation" and ADE de-activates the computer and exits.	

The following table covers the details of the per-transaction actions. These are the numbered steps (in red) as shown in the figure. Again, the flow is described in terms of one book, but there is no restriction in theory to fulfilling multiple books at one time. However, note that Content Server 4 will not support multiple fulfillments per transaction, though this support will be added in a future release.

Step	Action	Comment
1	User makes book selection and sends his payment and/or Distributor-specific authentication info to the Distributor.	
2	The Distributor requests authentication from the credit card/library authentication server, sending the payment and/or authentication info. Authentication server responds with authentication info.	Note that a distributor can be a book vendor or lending library. The workflows are almost identical.
3	Distributor responds to the browser request with the thank-you page. In the thank-you page, they embed the URL-encoded link (GBLink), which includes the information about the book to be fulfilled as well as any additional permissions	URL is signed (HMAC) by the Distributor
4	The user clicks on the thank you page, which causes the browser to invoke send the GBLink as a request to the server which is the target of the link	
5	The ACS server responds with the fulfillment token, which contains the unsigned license to the ebook. This token is handled by ADE.	
6	ADE then sends the aggregator the user's certificate	Note: ADE does cache the certificates, so this request need not be made each time
7	The Aggregator responds with their certificate, ensuring secure two-way communication.	
8	ADE then wraps the fulfillment token (containing the unsigned license for the book) in a fulfillment token. ADE then digitally signs the result with the user's private key and sends it to the Aggregator	

9	The Aggregator verifies the user's signature and the Distributor's signature (on the fulfillment token). If both are valid, it sends the fulfillment token to the Adobe license-signing service (Hancock) for signature, signing the request with the Aggregator's own signature. The signing service verifies that both the Aggregator's signature and the fulfillment token's signature are valid. If so, it extracts the license from the fulfillment token and signs it with Adobe's highly secure signing key. The resulting license is returned to the Aggregator.	Before the license is sent to Hancock it is filtered by the resource and distributor's rights, loan ID is inserted for loans and the result is merged with any existing licenses for that user
10	The signed license (wrapped in the fulfillment token) is returned to ADE in response to the request in step 8	
11	Using the media URL in the fulfillment token, ADE requests the bookbytes from the media server. The media server responds with the bookbytes.	
12	ADE writes the digitally signed license into the document. The resulting document is saved to disk and the local ADE manifest updated.	
13	ADE sends a "fulfillment complete" notification to the Aggregator.	

2 System Requirements

The system requirements are actually very generic. ACS4 was purposefully designed to be robust with respect to system requirements. The following requirements detail the system parameters under which Content Server 4 was validated. It is quite feasible to operate Content Server 4 on comparable components (JBoss instead of Tomcat, for example).

Operating System

- Linux®: Ubuntu 8.04
- Windows Server® 2003

Hardware:

- Processor: 32 or 64 bit

Runtime Memory:

- Services typically need 1 GB to run – very large documents may need more
- Java needs to be allocated a heap of at least 1 GB – more if processing large documents

Disk space:

- Services and Assets: Less than 50 MB for the services and assets.
- Plus approximately 2k per fulfillment for transaction logging

Database

- SQL. Microsoft® SQL Server, Oracle and MySQL are supported.

Objects stored in client database

- One object per document are ~ 1 kB in size and include
- Decryption key
- License token (~ 1 kB in size)
- Unique identifier

Application stack

- Java 1.5 or above (Note: Java 1.4 is NOT sufficient. Must be 1.5 or higher)
- Servlet engine (Tomcat has been used for validation but others should work)
- PHP 5 – required for the hosting the sample store – not Content Server 4 per se.

Communication

- Requires SSL client (for communication with signing server)
- Required bandwidth dependent on fulfillment rate, but per fulfillment traffic (all XML) with client comprises
    - Security certificates, ~ 1 kB
    - Fulfillment token, ~ 1 kB
    - License sent for signing is ~ 1 kB

3 Deliverables

The deliverables for ACS consist of a single archive (zip) file, which comprises the following contents:

Section	Item	Filename	Comment
Packaging			
	Packaging WAR File	packaging.war	
	Configuration file	`packaging-conf.txt`	This is located external to the war. See the installation instructions in section 4.5.2
Fulfillment			
	Fulfillment WAR File	Fulfillment.war	
	Configuration file	`fulfilment-conf.txt`	This is located external to the war. See the installation instructions in section 4.5.2
	Admin Console	admin.war	Comprises the HTML and the t form the Admin Console application itself, as well as the WebServices for accessing the fulfillment persistent objects
	Configuration file	`admin-conf.txt`	This is located external to the war. See the installation instructions in

			section 4.5.2
Migration			
	ACS3Convert	acs3toacs4convert.zip	The ASP/JavaScript files and updated DLLs that comprise the migration utility
Documentation			
	Content Server 4 User Manual	ContentServer_User_Manual.pdf	The overall user manual that provides and overview, manifest, installation guide and other information.
	Content Server 4 Technical Reference	ContentServer_Technical_Reference.pdf	The more detailed technical reference with schema and more detailed information
	Release Notes	ContentServer_Release_Notes.pdf	Release notes for the current release.
Sample Code			
	Packaging Sample	uploadtest-1.0.zip	Series of Java files, Eclipse project files and the resulting jar-file.
	Sample Books	samplebooks.zip	10 PDF and 10 EPUB files obtained from Gutenberg and converted to PDF and EPUB by Adobe
	Sample Store	samplestore.zip	Archive that contains the PHP and other sources that comprise the sample store.

4 Installation

4.1 Service installation overview

As noted in the sysem requirements, Content Server 4 supports both Linux (Ubuntu) and Windows Server (2003). This section details the general procedure for installing and setting up Content Server 4. Since a customer's actual setup may not precisely match that on which Content Server 4 was validated, the details listed below may vary in some respects. However, as Content Server 4 is very generic in its implementation, these guidelines should be very close.

The Content Server 4 services are designed to be able to all be placed on the same machine or on multiple machines. Any operability amongst different services is based on links found in the preferences as specified below. Each service requires a web server installed that is able to serve java servlets. A single web server may serve multiple services, but if the services are spread amongst multiple servers, each server must have a web server to run that service.

Example setups:

1. All on one server (as shown in the examples)
    - Main Server

- MySQL
- Tomcat
- Fulfillment Service
- Packaging Service
- Media Service (served by Tomcat)

2. Integrated Server plus Media Server (with network shared folder)
   - Content Server 4 Server
     - MySQL
     - Tomcat
     - Fulfillment Service
     - Packaging Service
   - Media Server
     - NFS accessible folder for loading files (packager)
     - Web server for distributing files
3. Integrated Server plus Media Server
   - Content Server 4 Server
     - MySQL
     - Tomcat
     - Fulfillment Service
     - Packaging Service
   - Media Server
     - Ftp Server for loading files (packager)
     - Web server for distributing files

In this document we will specify setting up a system from scratch, including a Tomcat server and a MySQL database. If these or an appropriate substitute already exist on your server, ignore these steps and use the paths applicable to your particular setup.

4.2 Setting up the applications

4.2.1 Overview

Prior to installing Content Server 4 you must first have installed an application server and a database server. If you do not already have these installed on a server, the general steps for doing this are:

- Install Java SDK
- Install Application Server
- Install Database Server
- Create "adept" Database

The remainder of this section provides the specific steps for installing Tomcat as an application server and MySQL as a database server on both Linux and Windows. If you decide to use other server technology such as JBoss or Microsoft SQL Server you will need to consult their reference material for appropriate installation steps.

4.2.2 Linux

Note: This installation guide is written for installing Content Server 4 on Linux with the users using a Bash shell. If you are using a different shell the syntax will need to be adjusted accordingly.

- Get a Java SDK 1.5. can be downloaded with following command:

```
sudo apt-get install sun-java5-jdk
```

  - The heap needs to be set to at least 1024 MB for packaging to run correctly.
  - Set an environment variable JAVA_HOME to the pathname of the directory into which you installed the JDK release. If you use this command, it will be:

```
export JAVA_HOME=/usr/lib/jvm/java-1.5.0-sun
export JRE_HOME=/usr/lib/jvm/java-1.5.0-sun
```

- Get Ant version 1.6.*

  http://ant.apache.org/bindownload.cgi or through apt-get:

```
sudo apt-get install ant
sudo apt-get install ant-optional
```

- Get Tomcat version 6.x
  - Get a source code distribution (either tar or zip) from http://Tomcat.apache.org/download-60.cgi
  - Untar/Unzip Tomcat into directory a temp directory, and run:

```
cd (TEMPDIR)
sudo ant download
sudo ant
cd output/build
sudo mkdir /usr/local/share/Tomcat/6.0.16/
sudo cp -R * /usr/local/share/Tomcat/6.0.16/
```

  - add the classpath to .bashrc for the user who will be running the service
  - Using your favorite editor, edit *~/.bashrc* and add the following line to the end of the document

```
export CATALINA_HOME=/usr/local/share/Tomcat/6.0.16/
```

  - Source this file to make it active now

```
source ~/.bashrc
```

- Get MySQL
  - http://ant.apache.org/bindownload.cgi or through apt-get:

```
sudo apt-get install mysql-server
```

  - When asked for a root password, enter the password you were sent in the welcome email from Adobe.
- Create the adept database

```
mysql -u root –p
CREATE DATABASE adept;
```

4.2.3 Windows

- Get a Java JDK 1.5.X
    - http://java.sun.com/javase/downloads/index_jdk5.jsp

  Note: you should give Java at least 512 MB of heap space, preferably 1 GB, e.g.

```
java –Xmx1024M
```

- Get Tomcat version 6.x
    - Download and install the Windows Service Installer from http://tomcat.apache.org/download-60.cgi
    - Heap size should be entered on the Tomcat config panel (Java tab). At least 512 MB is recommended.
- Get a SQL database client
    - For instance MySQL at http://dev.mysql.com/downloads/
    - When asked for a root password, use you were sent in the welcome email from Adobe.
- Create the adept database/schema. Create a database/schema named "adept" using one of the following procedures:
    - Using the command line tool, run the command

```
CREATE DATABASE adept;
```

- Or,
    - Using the Query Tool, right click in the "Schemata" section and select "Create New Schema". Enter "adept" and click "Ok"

4.3 Media Server

The media store can be a separate server from the rest of the servers. In order to use this service two pieces of information are needed: a base folder that files will be stored to and a base URL to retrieve the books from that server. This will be used as the root path for books which are added to the system. These preferences will be set in the packaging server

4.3.1 Example

If the baseLocation is "/share/books" and the baseURL is "http://mediaserver" then when the book "foo.epub" is added it will have the path "/share/books/foo.epub" and will be available at http://mediaserver/foo.epub. For more detail on this, see Upload and Download Locations in the Packaging section.

4.3.2 baseLocation

In order to upload a book to the media server a path will need to be provided. It may be one of the following forms:

- Local path "/share/books"
    - The simplest way to use this would be for a webserver to serve this folder directly
    - Another possibility is that a cron job can move the files from this path wherever is needed for the web server
- Network path "\\mediaserver\mediapath\books"
- Ftp URL: ftp://user:pass@mediaserver/path

4.4 Firewall exceptions

The following services must be accessible to the internet. All other services should be internal only.

4.4.1 Fulfillment Server

Links based on the serviceURL in web.xml of the fulfillment WAR file

- GBLink - http://<fulfillment server>/fulfillment/URLLink.acsm
- Fulfillment - http://<fulfillment server>/fulfillment/Fulfill
- Fulfillment Notification - http://<fulfillment server>/fulfillment/FulfillmentNotification
- Loan Return - http://<fulfillment server>/fulfillment/LoanReturn
- Auth - http://<fulfillment server>/fulfillment/Auth
- GetLicense – http://<fulfillment>/fulfillment/GetLicense
- ListResources – http://<fulfillment>/fulfillment/ListResources

4.4.2 Media Server

- HTTP port for downloading books as defined in PackagingPrefs com.adobe.adept.packaging.baseURL

4.5 Configuring the war files

Prior to deployment you must configure the packaging and fulfillment WAR files, to provide location of the database server set up in section 4.2 as well as customize other useful settings. The configuration of the WAR files can be done on any system.

4.5.1 Get the war files fulfillment.war and packaging.war

These files are provided as part of the deliverable.

4.5.2 Property files location

In most cases, the configuration files will be on your local drive in one of the following locations:

- On Windows, the configuration files will be in the c:\config folder:

```
c:\config\fulfillment-conf.txt
c:\config\packaging-conf.txt
c:\config\admin-conf.txt
```

- On Linux systems, the configuration files will be in the /etc folder

```
/etc/fulfillment-conf.txt
/etc/packaging-conf.txt
/etc/admin-conf.txt
```

These locations are determined by a configuration setting inside the war file and can be changed.

```
com.adobe.adept.config1=c:/config/fulfillment-conf.txt
```

Any properties specified in these external files will override the same properties specified in the config file in the WAR file itself.

Note that these configuration settings can also be changed inside the WAR file itself, which allows different settings for various environment settings such as in the development stage (i.e. by setting the “real” settings in the WAR file and then overriding those settings with a configuration file For each war file, Unzip <foo>.war (It may be useful to temporarily change the extension to ".zip", just remember to change back to ".war" when done).

Property files are located in the directory “WEB-INF/classes” inside the WAR file. At the base level of this directory may be the following properties files. These files hold properties specific to certain stages of development. If the file you want does not exist, add that file. The stage is read by looking at the ENVIRONMENT_NAME system property.

The key properties file are:

- *properties.txt* - This file will always be read, its values may be overridden by later files. Any stage-generic properties should go here.
- *properties_dev.txt* - This file will only be read if the system is in the DEBUG or DEVELOPMENT stage.
- *properties_stg.txt* - This file will only be read if the system is in the PRE_STAGING or STAGING stage.
- *properties_prd.txt* - This file will only be read if the system is in the PRE_DEPLOYMENT or DEPLOYMENT stage.

Note that it is also possible (and sometimes more convenient) to set these properties through the standard Java system properties.

4.5.3 Common properties settings

- Database Connections
  Make sure that each of the com.adobe.adept.persist.sql.* properties matches the database setup. If you installed using the default setup, this following example will work. If not, please set the URL to the correct address and make sure that the SQL server is allowed to be administered by this box:

```
com.adobe.adept.persist.sql.driverClass=com.mysql.jdbc.Driver
com.adobe.adept.persist.sql.connection=jdbc:mysql://127.0.0.1:3306/adept
com.adobe.adept.persist.sql.user=root
com.adobe.adept.persist.sql.password=******
com.adobe.adept.persist.sql.dialect=mysql
```

- Logging Information
  com.adobe.adept.log.level determines which messages are written out to the log, based on Log4j levels. Some useful values for this (in decreasing levels of logging severity) are "trace” and "error". One can also turn logging “off”, but this is strongly discouraged. The overhead is very small and the information is quite valuable.

```
com.adobe.adept.log.level=trace
```

- com.adobe.adept.log.file specifies the path to the log file to be written

```
com.adobe.adept.log.file=/var/log/fulfillment.log
```

- Other logging settings (The default values are below. These properties only need to be present if you wish to override the default):

4.5.4 Setting up the Logging

- com.adobe.adept.log.pattern=<Pattern in log4j pattern layout>
- com.adobe.adept.log.maxSize=<Maximum size of log file before being rolled to a backup file, for instance "1MB">
- com.adobe.adept.log.maxBackupIndex=<Maximum number of backups to keep. These backups are created once the log file goes over the maximum size specified above.>
- com.adobe.adept.log.bufferedIO=<true or false to determine whether the logging IO should be buffered>

```
com.adobe.adept.log.pattern=d{DATE} %-5p [%c@%t]: %m%n
com.adobe.adept.log.maxSize=1MB
com.adobe.adept.log.maxBackupIndex=5
com.adobe.adept.log.bufferedIO=false
```

4.5.5 Fulfillment settings

You need to set the following property to say which signing server is used for signing

```
com.adobe.adept.fulfillment.security.licensesignURL=
https://nasigningservice.adobe.com/licensesign
```

or to https://eusigningservice.adobe.com/licensesign depending upon your locale.

```
com.adobe.adept.serviceURL = http://.../fulfillment
```

Note that the service URL should point to location of your Content Server 4 server (the URL that you supplied to Adobe and is contained in the P12 file). So the URL is that URL with "fulfillment" appended (since the name of the fulfillment WAR file is *fulfillment.war*).

4.5.6 Packaging specific

The packaging server will place the files onto the Media Server. This server can either be the local box or a remote server.

The baseLocation preference is the default location for books to be placed on the server. This can either be a path or a FTP URL ("ftp://<remote host>/<remote path>")

The baseURL preference specifies what URL shall be used to download the book from the location stored in the baseLocation.

```
com.adobe.adept.packaging.baseLocation=/usr/local/share/Tomcat/6.0.16/webapps/ROOT/books
com.adobe.adept.packaging.baseURL=http://<DNS name for Media server>/books
```

The service URL for packaging should be:

```
com.adobe.adept.serviceURL = http://hostname/packaging
```

Note that the service URL essentially points to packaging WAR file, so the name is the combination of the hostname of the computer on which it is hosted and the name of the WAR file, i.e. packaging (since the packaging war is named *packaging.war*).

4.5.7 table_list.txt

The file *table_list.txt* is found at *<unzipped <foo>.war>/WEB-INF/lib/<unzipped <foo>.jar>/table_list.txt* (for instance *fulfillment.war/WEB-INF/lib/fulfillment.jar/table_list.txt*). This file contains a list of all the tables referenced from the database by this code and is used to ensure that all tables are created upon initialization.

4.6 Installing the services

To install the services onto a Tomcat install, simply copy the appropriate WAR files into the webapps directory. If following the instructions above, this directory will be */usr/local/share/Tomcat/6.0.16/webapps/* (Linux) or *C:\Program Files\Apache Software Foundation\Tomcat 6.0\webapps* (Windows). Once the web service is started, this WAR file will be automatically handled by the server, no further configuration needs to happen.

4.7 Starting the system

4.7.1 On Linux

The user which runs Tomcat should have write access to the log directories, both for the services as defined above and for Tomcat itself (*/usr/local/share/Tomcat/6.0.16/logs/* in this example)

- Starting the service running:

```
sudo /usr/local/share/tomcat/6.0.16/bin/catalina.sh start
```

- Stopping the service:

```
sudo /usr/local/share/tomcat/6.0.16/bin/catalina.sh stop
```

4.7.2 On Windows

Tomcat should start automatically. It may be manually started or stopped either using the Administrative Tools:Services Control Panel or by right clicking the task bar icon and choosing the appropriate task.

4.8 Verifying the system is running

Running Status Check will tell you what is correctly operating on this server. If the page does not load, check to make sure Tomcat is running properly as described below. If the page loads and all items have check marks, you can ignore the rest of this section, otherwise you should check the section which is failing to isolate the problem.

- http://<fulfillment servername>/fulfillment/statuscheck
- http://<packaging servername>/packaging/statuscheck

4.8.1 Checking Tomcat directories

- Go to the webroot folder

```
cd /usr/local/share/tomcat/6.0.16/webapps
```

  - Check for a folder to go with each .WAR file (check to make sure "fulfillment" and "packaging" exist)
  - Check to make sure there is a WEB-INF folder in the root directory of each of each directory made from a WAR file.

4.8.2 Checking MySQL

- Start mysql and login

```
mysql -u root -p
```

- Select the adept schema

```
use adept;
```

- Check the list of tables. The system should have created the tables found in the table_list.txt on initialization

```
show tables;
```

4.9 Known Issues

4.9.1 Hosting the store

Tomcat is needed on the fulfillment server, but there are some known conflicts between Tomcat and PHP5. We suggest you host the store on a different server than where the fulfillment server is hosted. We are investigating the issue.

5 Database Setup

5.1 General

Content Server 4 is written in Java. It uses the standard Java Database Connectivity (JDBC) APIs to communicate with the database. Most database providers (including MySQL, Microsoft and Oracle) have JDBC drivers available to communicate with their database. To identify a particular database and to authenticate the connection JDBC uses three pieces of information:

- JDBC connection string (which includes database type, database server/port and database name)
- user and password.
- JDBC driver class name is needed to load and register it.

Each database vendor supports a slightly different set of features and uses slightly different SQL syntax to expose them. Content Server 4 built-in persistence layer needs to know which SQL dialect should be used for communication.

Notes

1. **IMPORTANT:** Adobe Content Server 4 expects and requires that the database allow for transactions, and in particular rolling back of transactions. Unfortunately the default installation of MySQL on all systems sets the default storage engine for tables to be "MyISAM" which is not a transactional storage engine. We have tested and validated with the "InnoDB" storage engine and strongly recommend using that. If you do not, you can expect to encounter database consistencies problems.

   All MySQL users should carefully check which storage engine they are using via the MySQL Configuration manager.

   If you find that you have MySQL configured to use MyISAM, you need to fix this. To do so, you must either ALTER your existing tables, or set the default storage engine to be "InnoDB" and drop all tables in your current schema – whichever is easiest.

   To ALTER existing tables the syntax is:

   ```
   ALTER TABLE  '<schema>'.'<tableName>' ENGINE=InnoDB;
   ```

   where <schema> is replaced with your schema name and <tableName> is replaced with the table name.

   To set the default storage engine to be InnoDB you must add the following line into the [mysqld] section of your /etc/my.cnf file:

   ```
   default-storage-engine=innodb
   ```

   Note: You must restart MySQL after changing this setting!

2. The tables in the database are set up automatically by Content Server 4. All Content Server 4 needs to have is access to the database as detailed below.

3. While processing a fulfillment the transaction ID in the incoming request is checked to see if it has already been processed. If the transaction IDs are meant to be unique by case (i.e. case sensitive), and the database is set up to make case insensitive matches, there may be erroneous matches returned, causing the fulfillment to fail. Please make sure that your transaction IDs are unique regardless of case, or that you have the database setup to do case sensitive matching on queries.

5.2 JDBC Driver

The JDBC driver is simply a jar file. That jar file needs to be placed where Java can find it, e.g. in the app server lib folder. It also can be placed in the WEB-INF/lib folder inside war file – but if that approach is used, then that step has to be done for both packaging and fulfillment services war-files.

5.3 MySQL

The JDBC driver is called "Connector/J"; information about it is available at http://www.mysql.com/products/connector/j/

Sample properties:

```
com.adobe.adept.persist.sql.driverClass=com.mysql.jdbc.Driver
com.adobe.adept.persist.sql.connection=jdbc:mysql://127.0.0.1:3306/adept
com.adobe.adept.persist.sql.user=root
com.adobe.adept.persist.sql.password=XXX
```

5.4 Microsoft SQL Server

JDBC driver info is at http://msdn.microsoft.com/en-us/data/aa937724.aspx

An important note on Microsoft SQL server set-up: connecting from Java was only tested using "mixed" authentication mode (that uses standard username/password). It is not clear if JDBC supports (default) Windows authentication mode at this point. Also, connecting from Java requires TCP/IP connection to a particular port number. In the management utility, TCP/IP has to be enabled and "TCP Dynamic Port" feature disabled (corresponding setting is located on "IP Addresses" tab on TCP/IP properties; all fields for dynamic ports have to be blank (not zero!) and "TCP Port" fields set to 1444 – or whatever port you want to use).

Sample properties:

```
com.adobe.adept.persist.sql.driverClass=com.microsoft.sqlserver.jdbc.SQLServerDriver
com.adobe.adept.persist.sql.connection=jdbc:sqlserver://localhost:1444;databaseName=tempdb
com.adobe.adept.persist.sql.user=sa
com.adobe.adept.persist.sql.password=XXX
com.adobe.adept.persist.sql.dialect=microsoft
```

5.5 Oracle

Oracle has two different JDBC drivers. Only the "thin" one was (very lightly) tested as of the initial release.

See http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html

Sample properties:

com.adobe.adept.persist.sql.driverClass=oracle.jdbc.driver.OracleDriver
com.adobe.adept.persist.sql.connection=jdbc:oracle:thin:@//dbhost:1521/XE
com.adobe.adept.persist.sql.user=root
com.adobe.adept.persist.sql.password=XXX
com.adobe.adept.persist.sql.dialect=oracle

5.6 Database Schema

When Content Server 4 connects to the database for the first time, it checks if the table “schemaversion” exists, and if not, it creates it. Then it will check that table for the schema version for each table that Content Server 4 needs. If there is no entry for that table, Content Server 4 will create that table. If the schema is out of rev, Content Server 4 will try to update the table’s schema. Thus, the connection to the appropriate database is all the setup that Content Server 4 requires.

Here is the list of tables that Content Server 4 reads and writes:

distributionrights
distributor
distusednonce
fulfillment
fulfillmentitem
license
licensecertificate
resourceitem
resourcekey
schemaversion
secondaryresource
userpublic
userusednonce

6 Content Server 4 Operator Enrollment

6.1 What the Operator Enrollment Process Does

License signing system usage is tracked by the request URL. Similar to SSL, Adobe validates requests using PKI certificates and keys. The operator enrollment process allows the operator (you) to create a new set of keys and request a certificate from Adobe that associates the given public key with a particular URL (and the operator). This also lets Adobe know that you are using a particular URL. The enrollment process consists of 4 parts:

- Creating keys
- Creating a Certificate Signing Request
- Sending the CSR to Adobe

- Associating the certificate with the private key (creating a P12 file).

After you have created your P12 file you will want to configure the fulfillment server to use the new P12 file.

6.2 Preparation

6.2.1 Java

Windows

Download from JRE 1.5 for Windows from http://java.sun.com/products/archive

Mac

Java is preinstalled.

Linux

This will be specific to most distributions. Please consult your admininstrator.

6.2.2 OpenSSL

Windows

A good overview about installing OpenSSL with Cygwin can be found at http://daveonsoftware.blogspot.com/2007/06/installing-cygwin-with-openssl-package.html. It has instructions for setting up OpenSSL with Cygwin - OpenSSL won't be installed by default. You can get Cygwin from http://www.cygwin.com/setup.exe.

Mac and Linux

OpenSSL is preinstalled.

6.2.3 operatorClient

Get the operatorClient executable from the Content Server 4 zip file.

6.3 Creating the Keys and Certificate Signing Request

The License Signing service needs a certificate signing request in PKCS10 format. The following line uses OpenSSL to create an RSA KeyPair and a CSR in PKCS10 format:

```
openssl req -newkey rsa:1024 -out operator.pkcs10 -keyout operator.key
```

You then need to answer questions on the command line - it doesn't really matter what you enter as those values will be overwritten in the certificate creation process on the License Signing server.

So, as an example, one might see this, where to the left of the colon is the SSL prompt and to the right is what you enter:

```
Country Name (2 letter code) [AU]: US
State or Province Name (full name) [Some-State]: California
Locality Name (eg, city) []: San Jose
Organization Name (eg, company) [Internet Widgets Pty Ltd]: Adobe
Organizational Unit Name (eg, section) []: Digital Publishing
Common Name (eg, YOUR name) []: Ric Wright
Email Address []: digitaleditions@adobe.com
```

You will then see the following prompt and two more questions. DON'T answer them. They are not needed, and OpenSSL has been known to crash if you do answer them!

```
Please enter the following 'extra' attributes to be sent with your certificate request.
A challenge password []:
An optional company name []:
```

The file *operator.pkcs10* will contain certificate request (to be uploaded to Adobe) *operator.key* will contain the password-protected private key.

Adobe will sign your certificate request and provide you with the certificate file *operator.cer*

6.4 Sending the CSR to Adobe

To send a CSR to Adobe you will need to use the "operatorClient.jar" Java program that is inside the .zip file.

6.4.1 Login

NOTE: For our non North American customers please change the "License Signing URL" to https://eusigningservice.adobe.com/licenseadmin and click on the "Save" button before clicking on the "Log In" button. Failure to do so will cause errors in later steps.

Start operatorClient and then login as prompted with the AdobeID and password that you were sent after your purchased Adobe Content Server 4.

6.4.2 Operator URLs

Enter the URL that you will be using for your fulfillment server. If the port you are using is 80 for HTTP, 443 for HTTPS), you may omit the port. If you are using a different port, the value of the port must be specified as well. The URL must include the full path to the fulfillment service e.g.: http://www.example.com/fulfillment and not http://www.example.com/

NOTE: Using 'localhost' might seem convenient but is strongly discouraged. It can work in some circumstances (e.g. with some ports and with some parts of the system) but won't in others. So for these reasons its use is strongly discouraged.

6.4.3 6.4.3 Submitting request

Select the CSR (PKCS 10 file) that you created in section 6.3 and select an output location for the X.509 public key certificate you will use in the following steps.

6.5 Creating the P12 file.

6.5.1 Convert operator.cer from DER to PEM

OpenSSL p12 packaging requires the certificate to be in PEM format. The LicenseSigning service returns the certificate in DER format. To convert from DER to PEM use the following OpenSSL command

```
openssl x509 -in operator.cer -inform DER -out operator.pem -outform PEM
```

6.5.2 Creating the P12 file

Assuming you named the output file *operator.pem* then run this command to package the certificate and the key together:

```
openssl pkcs12 -export -in operator.pem -inkey operator.key -name operator –out
operator.p12
```

This command will prompt you for the "export password". This password is used to protect your p12 file, and will need to be used to configure the Content Server 4 installation.

6.6 Configuring Fulfillment Server to use P12 file

In the config file you will want to have the following lines

com.adobe.adept.fulfillment.security.keystore=pkcs12

com.adobe.adept.fulfillment.security.pkcs12.file= … URL to location of p12 file created in step 6.5.2 …

(i.e. file:///c:/operator.p12 and not c:/operator.p12)

com.adobe.adept.fulfillment.security.keystore.password= …export password from step 6.5.2 …

7 What's Changed from Content Server Version 3 to 4

7.1 Based on Headless Services

One of the primary changes from Content Server 3 to Content Server 4 is that where Content Server 3 presented implementers with a "turnkey" style solution, Content Server 4 has been designed from the beginning as a set of headless services that implementers can integrate easily into their existing systems.

There are two core services:

- The packaging service provides for preparing documents to be fulfilled by Content Server 4. This includes the encryption, setting permissions and adding metadata to the documents.
- The fulfillment service provides license signing and management of the whole license process.

These two core services, which are packaged as separate WAR files, comprise the essential and required parts of Content Server 4. In addition to these two core services, Content Server 4 also comprises a number of pieces which provide examples of how an implementer can integrate these services into their own infrastructure. For each of these pieces (in contrast to the core services), Adobe has provided the actual source code so that implementers can both see a clear example of how they can integrate with Content Server 4 as well as pieces that they may wish to customize to fit their infrastructure.

Content Server 4 does not share any actual code with the previous implementation - with the exception of leveraging some of the facilities in the existing Content Server 3 infrastructure to support the migration of existing Content Server 3 content to Content Server 4. The core of the new Content Server 4 implementation is written in Java. The services themselves are exposed as web services. The persistence objects can be accessed directly from Java, but all other services are available only via web services. Both the web services and persistence objects are covered in more detail later in this document.

One important aspect of this design is that Content Server 4, in contrast to Content Server 3, does NOT have its "own" database. Instead, it has some tables within the existing infrastructure's own database. Content Server 4 doesn't know where the tables actually are, all it knows is how to write to its own data-persistence objects, which write via JDBC drivers into the database. Setting up the drivers is part of the integration process. How this should be done is covered in detail previously in this document.

7.2 License Signing

A key aspect of Content Server 4 is that the "voucher" of the Content Server 3 infrastructure has been replaced by the Content Server 4 license. The license-template for each book being hosted in the Content Server 4 system is stored in the database. When a book is going to be fulfilled, the template is wrapped in a "fulfillment token" and sent to the client

(Digital Editions). It is then filled out with some of the user's info and returned to the fulfillment service. The fulfillment service then sends the license to the Adobe-hosted license service where the license is digitally signed by Adobe's hosted service and returned. The digitally signed license is then returned to the client, where, when the bookbytes have been successfully downloaded, it is written into the resulting document. The client software, Digital Editions (and implementations of the same document engine on mobile devices such as the Sony® Reader) will only open documents containing a valid license that has been digitally signed by Adobe's hosted license-signing service.

7.3 Use of Digital Signatures

At several steps during the fulfillment process, digital signatures are used. Digital signatures are a very secure method of preventing tampering with information being passed between computers.

7.4 GBLink Changes

In order to ease integration of Content Server 4 into customer's existing infrastructure and workflows, Adobe has tried, insofar as possible, to preserve existing workflows. Many distributors (stores and library sites) use the GBLink method to trigger the fulfillment process. GBLink is supported by Content Server 4, exactly as Content Server 3 did, but with two differences:

- Only one book per URL is supported. Content Server 3 supported the embedding of multiple books per URL. As far as can be determined, this feature was/is little used. But support for this is not present in Content Server 4 because it cannot be guaranteed to always work since the length of a valid URL varies depending on the browser, proxy, etc. By limiting the number of books per URL, we can guarantee that the URL will not be truncated and all the information in the URL will come through correctly. Note that this is especially important on mobile devices where URL handling is even less predictable.
- The handling of the GBLink information was redesigned to avoid a current Content Server 3 problem: the need to cap the number of times a download link can be used. The consequence of the need to cap, or limit, the number of retries was that if the user did run into problems, the Content Server 3 operator often had to re-issue the download link. This problem goes away completely in Content Server 4. There is no need to cap the number of retries for a given link since there is no way for the user to misuse the download link (by giving it to someone else, for example). More information on this aspect is covered later in the document.

7.5 Incremental Permissions

Content Server 4 supports most of the same permissions that Content Server 3 does. Some of the more esoteric permissions of the EBX protocol (metered expiration, etc.) have been dropped since they were not fully supported anyway. Content Server 4 does support all the common, essential permissions including copy and print N excerpts or pages per a given time period. What has changed is that the permissions now refresh incrementally. Where in Content Server 3 if a user had the permission to print 10 pages every 10 days and they printed out 10 pages on day one, they would have to wait 10 days before they could print any more pages. In Content Server 4, this has changed so that if the user uses up all their 10 pages on day one, then on day two then can print one more page (since they effectively accumulate one incremental page per day). The net effect is the same, the only difference is that in Content Server 3 the accumulation of new pages was not incremental, while in Content Server 4 it is.

Also note that permissions in general have been re-designed to be more flexible and powerful. However, the permissions supported by Content Server 4 are for the most part a superset of the permissions supported by Content Server 3. This subject is covered in some detail in section 9, Digital Rights Management in Content Server 4.

One permission that has been dropped is support for constraining whether a book can be "Read Aloud". In Content Server 4 it is not possible to prevent a user-agent from allowing "Read Aloud".

7.6 Client Side Activation and Authorization

Like Content Server 3, Content Server 4 supports both anonymous and authorized (named) activation. Activation is the process in which a public/private keypair (required for the digital signature handshaking between the client and the Content Server 4 fulfillment service) is issued to each client machine (desktop PC or non-tethered mobile device). Authorization is the process of associating an Adobe ID with a particular machine. A single user can initially, at the time they first authorize their first machine, authorize up to 6 machines and 6 tethered devices. However, since it was recognized that machines get "used up" over time (get destroyed, sold or otherwise disposed of), the allowable number of machines that a single individual can authorize slowly grows over time. Specifically, the user can authorize one additional machine and one additional tethered device per year.

8 PDF and EPUB Formats

Content Server 4 supports packaging and fulfillment of both PDF and EPUB. The details of what is supported are covered in the following section and in more detail in Appendix 2

8.1 PDF

In general, PDF 1.6 is supported, with the exceptions listed below. Note that many of these exceptions will not cause problems during packaging and fulfillment, but the Digital Editons client software cannot render them so they are effectively not supported. For example, it is perfectly feasible to package a file that has JavaScript and interactive forms, but this file will not render correctly in Digital Editions. On the other hand, some of the other features will cause errors on the server side, e.g. files protected by non-Content Server 4/EBX DRM schemes.

The following PDF features should not prevent packaging, but will not render properly in Digital Editions.

- Interactivity (i.e. JavaScript or interactive forms)
- Forms (both old and new formats)
- JPEG2000 is not supported in the current release (but will be in the next)
- Some advanced transparency features (e.g. soft masks, non-separable blend modes, knockout groups) are not supported in the current release of Digital Editions (but will be in the next release).
- PDF Packages
- Digital signatures
- For PDF annotations, only link-style annotations are supported. All others, include 3D annotations, are not and will not be rendered properly if at all
- Some embedded multimedia (e.g. QuickTime.)

The following PDF features should cause the packager to fail and return and error message.

- Password-protected documents (i.e. not EBX or Content Server 4 DRM)
- Certificate Security (pubsec) documents
- Adobe LiveCycle® Policy Server (APS) protected documents
- Third party DRM solutions (e.g. FileOpen)

8.2 EPUB

The EPUB support is similar to PDF support in that it will package any file it can open. However, the client may not render every EPUB, but it will render almost any valid EPUB. Digital Editions implements pretty much the full EPUB

spec, which can be obtained from the IDPF site. Only a few fairly esoteric error handling strategies and the full DTBook syntax are not supported (DE does support a subset of the DTBook syntax).

9 Migrating Books from Content Server 3

9.1 Introduction

If you are an Content Server 3 operator, you will presumably have a repository of books that you wish to migrate to Content Server 4. The Content Server 3 migration tool is provided for this purpose. In essence, this is a script that works with your existing Content Server 3 and copies the entries for the Content Server 3 packaged and deployed content from the Content Server 3 database into the Content Server 4 database so that the Content Server 3 content may be fulfilled through the Content Server 4 server. You do NOT have to repackage the books themselves. The tool accesses the Voucher table in the Content Server 3 database and for each voucher in the database; the content (decryption) key as well as the book metadata and deployment information are extracted and sent to the Content Server 4 database. When the migration completes the Content Server 3 content will be available to be fulfilled from the Content Server 4 server.

ACS3Convert is a Microsoft IIS ASP web application, and installs and runs alongside the existing Content Server 3 IIS Web applications. It uses shared ASP code from the ACSCommon web app area of the Content Server 3 installation.

ACS3Convert communicates to the Content Server 4 server via the standard Content Server 4 web services for Manage ResourceKey and Manage ResourceItem.

ACS3Convert uses a database connection to track the status of the Content Server 3 vouchers. Once a voucher record has been fully processed, and the relevant information has been sent to the Content Server 4 database, that voucher is marked as complete and is not subject to any further processing by ACS3Convert. This will allow Content Server 3 vouchers that cannot be immediately processed to be tracked in the ACS3Convert database so that the Content Server 3 item can be appropriately marked and then only items that have not already been transferred to Content Server 4 will be processed in subsequent runs of ACS3Convert.

ACS3Convert also writes a log file in CSV format with the results of a processing run to provide an accurate record of the ACS3Convert processing runs.

Finally, ACS3Convert can be invoked in two ways. It can be simply started, in which case it will attempt to migrate the entire Content Server 3 Voucher table contents. Alternatively, it also accepts a text file with a list of vouchers as input to enable specific conversion attempts. This is particularly useful where Content Server 3 items failed to be migrated in the previous run of the migration utility.

For further information on migrating your existing Content Server 3 content to version 4, please consult the ACS3_to_ACS4_Conversion_Utility document that is packaged with the migration utility itself.

10 Digital Rights Management in Content Server 4

10.1 Permissions

Content Server 4 permissions are expressed in XML within a permissions element. The permissions element may contain any number display, excerpt, or print elements, which may in turn contain one of each type of a valid constraint (see below for list) element for that permission element. In order for a permission to be granted it must be present in the permissions element. If there are no permissions element of a given type (i.e. no print elements), then that permission is not allowed. Finally, note that multiple permissions of the same type are allowed and that for each set of permissions, all constraints must be satisfied.

10.2 display

The display permissions control whether and how the user can actually view the book.

Valid constraints are:

deviceType

Specifies the type of device that the display permissions apply to.

If a deviceType constraint is not specified, then the resource can be viewed on all device-types (subject to the other constraints)."

Possible values are:

- **standalone** - a personal computer that can be both on- and off-line
- **public** - a public terminal which is always on-line
- **roaming** - a public computer with roaming profile which is always on-line
- **mobile** - a mobile device that can be both on- and off-line and can be attached to another device
- **tethered** - a handheld device that cannot access Internet directly and can be attached to another device
- **storage** - a storage-only device that can only store data and can be attached to another device
- **token** - a fingerprint-only device that only provides unchangeable fingerprint and a small amount of storage for the activation info

Note: At present, the client software, Digital Editions 1.6, supports only standalone for Mac and Windows PC and tethered for the Sony PRS505A.

Specifying any other type will effectively preclude any permissions. However, specifying any other device type will not **today** give user additional permissions, but may do so in future. In particular, if you want certain permissions to be granted to all handheld devices, we recommend always adding additional permission elements for "mobile" deviceType which parallels elements with the deviceType set to "tethered".

.

duration

Controls how long a resource can be viewed after fulfillment. This is converted into an "until" constraint at the time of fulfillment.

until

If used, its value is an absolute date and time (e.g. Valid until December 31, 2009 11:00 pm EST).

While both an until constraint and a duration constraint can be specified prior to fulfillment, at fulfillment time this will result in a single until constraint which uses the nearer of the two resulting dates.

device

If specified, indicates that the resource may be used on only one device (the one to which the resource is fulfilled)

loan

This permission is set only during the fulfillment process. It simply contains the ID of the loan itself.

10.3 excerpt

Controls the ability to copy data from the resource to the clipboard. The constraints are similar in many respects to display:

deviceType

Specifies the type of device that the excerpt permissions apply to. Using the deviceType field determines whether copying is allowed on a specific type of device. The possible values are the same as above in section 9.2.

duration

Controls how long a resource can be copied from after fulfillment. This is converted into an "until" constraint at the time of fulfillment.

until

If used, its value is an absolute date and time (e.g. Valid until December 31, 2009 11:00 pm EST).

While both an until constraint and a duration constraint can be specified prior to fulfillment, at fulfillment time this will result in a single until constraint which uses the nearer of the two resulting dates."

.

count

Number of times an excerpt can be made. Count has three attributes:

- **initial** - initial allowance for the count, given when document is fulfilled (on one device only) (optional)
- **incrementInterval** - time in seconds, after which a new single copy operation is granted (optional)
- **max** - point at which permission will cease to be incremented until current permission count falls below that level" (optional)

While both initial and incrementInterval are optional, one of the two must be specified. If max is not included (since it is optional), there is no maximum count.

Note that in contrast to Content Server 3, copy (and print) operations accumulate singly over a specified period. So "10 copy-operations each 10 days" in Content Server 4 means that a new copy-operation is possible each day. The user begins with the count of **initial** and accumulates a single new copy-operation each **incrementInterval** seconds up to a maximum of **max** copy-operations.

loan

This permission is set only during the fulfillment process. It simply contains the ID of the loan itself. This permission gets created when the client detects that the duration or until elements have been specified.

10.4 print

Controls the ability to print from the resource. The constraints are similar in many respects to excerpt.

deviceType

Specifies the type of device that the print permissions apply to. Using the deviceType field determines whether printing is allowed on a specific type of device. The possible values are the same as above in section 9.1.1.

duration

Controls how long a resource can be printed from after fulfillment. This is converted into an "until" constraint at the time of fulfillment.

until

If used, its value is an absolute date and time (e.g. Valid until December 31, 2009 11:00 pm EST).

While both an until constraint and a duration constraint can be specified prior to fulfillment, at fulfillment time this will result in a single until constraint which uses the nearer of the two resulting dates."

.

count

The number of pages that can be printed. Count has three attributes:

- **initial** - initial allowance for the count, given when document is fulfilled (on one device only) (optional)
- **incrementInterval** - time in seconds, over which a single new unit of permission for printing a page is granted (optional)
- **max** - maximal count that can be accumulated, count is not incremented beyond that number (optional)

While both initial and incrementInterval are optional, one of the two must be specified. If max is not included (since it is optional), there is no maximum count.

Note that in contrast to Content Server 3, print (and copy) operations accumulate singly over a specified period. So "10 pages printed each 10 days" in Content Server 4 means that a new page-print is possible each day. The user begins with the count of **initial** pages that can be printed and one additional page can be printed each **incrementInterval** seconds up to a maximum of **max** pages that can be printed.

maxResolution

Determines the maximum resolution of the device (in dots per inch - DPI) to which the resource will be allowed to be printed. Note that if the maxResolution is lower than the actual resolution of the target printer, the output will be downsampled to the specified maxResolution.

loan

This permission is set only during the fulfillment process. It simply contains the ID of the loan itself. This permission gets created when the client detects that the duration or until elements have been specified.

10.5 Examples

The following are a few examples of the use of the permissions syntax. The complete RelaxNG Schema can be found in Appendix A.

Here is an example of allowing unlimited copying and printing and display on any device.

```
<permissions>
    <display/>
    <print/>
    <excerpt/>
</permissions>
```

This is an example of allowing display on any device, but not allowing any printing or copying at all.

```
<permissions>
    <display/>
</permissions>
```

Here is a more complex example.

```
<permissions>
    <display/>
    <excerpt>
        <device/>
        <until>expiration date</until>
        <count initial="10" max="20" incrementInterval="86400"/>
    </excerpt>
    <print>
```

```
        <device/>
        <maxResolution>150</maxResolution>
    </print>
    <print>
        <maxResolution>300</maxResolution>
        <count initial="10" max="20" incrementInterval="36000"/>
    </print>
</permissions>
```

The result of this set of permissions is:

- Display is allowed on any device
- Copying is allowed on a single device until the expiration date is reached, starting with 10 copy-operations and accumulating a new copy-operation every day (86400 seconds). A maximum of 20 copy-operations can accumulate.
- Unlimited printing at 150 DPI or below is allowed.
- Printing in resolution of above 150 DPI and below 300 DPI is allowed, starting with 10 pages initially, accumulating a new page 10 hours, up to a maximum count of 20 pages.

11 GBLink Support in Content Server 4

11.1 Overview

In Content Server 3, the primary method for the user to trigger the start of the fulfillment process was to click on a URL that contained the parameters for the fulfillment. The link was also digitally signed (with HMAC). To ease integration for existing Content Server 3 operators, Content Server 4 has been designed to support existing GBLink format without any changes required by the distributors. These "GBLink" URLs can be embedded in either webpages or HTML-compatible emails.

This section provides details of how the URLs are constructed and what parameters are supported.

11.2 Sample Content Server 3 GBLink URL

Here is a sample Content Server 3 GBLink. This sample is broken at the query string separators for legibility. Note that Content Server 4 does not support some of the specified parameters (more on this in the next section).

```
http://fulfill.acme.com/fulfill/ebx.etd?
action=enterorder&
ordersource=gbdev&
orderid=DYN-gbdev-851848657369052&
handoff=true&
bookid=ISBN%3A0038910561&
price=0&
rights=$uat%231209587872$$cpy%235%23900$$prn%235%23900$&
gbauthdate=4%2F30%2F2008+19%3A40+UTC&
dateval=1209584435&
gblver=3&
auth=933d01cb6cb561d7f9eb1029826b73927b90a29d
```

11.3 Differences between Content Server 3 and 4 Support

As noted previously, Content Server 4 will support current GBLink syntax and format without any changes required by the distributor. However, while Content Server 4 will consume and process the existing ACS3GBLink-style URL, there are a few differences in what gets processed. First and foremost, Content Server 4 does **NOT** support multiple resources per URL. Only one resource (e.g. book) can be encoded in a given URL. This is different from Content Server 3, which allowed an unlimited number of resources per URL. This feature was little used and could not always be counted on to work since browsers and proxies vary widely in how long a URL they support. For this reason, support for this feature

was dropped from Content Server 4. Instead, Content Server 4 supports multiple resources per fulfillment token. To use that feature, distributors would need to generate the fulfillment token themselves (See Appendix B).

The second difference in GBLink handling in Content Server 4 is that some of the parameters are simply ignored by Content Server 4. The following table lists the parameters for GBLink and which are supported in each version.

Required Parameter	Value	Content Server 4	Comment
action	"enterorder" for purchase, "enterloan" for loan	Yes	
ordersource	The key name that identifies the Distributor as recorded in the Distributor's entry in the Content Server 4 database	Yes	
orderid	A unique string identifying this order to the fulfillment server.	Yes	
handoff	true	No	
bookid	The book identifier or SKU as recorded in the eBook database. The Content Server 4 Fulfillment service searches first for a book identifier matching this value and then for a SKU. A book identifier must have the form *id_type*:*id_value*; a SKU can have any form. Example: ISBN:1936014010	Yes	Note that Content Server 4 does NOT support multiple books per URL
resid	Content Server 4 resource ID	Yes	
gbauthdate	The date and time of the order, in the following form: mm/dd/yyyy hh:mm **UTC** The Adobe GBLink service generates this value. This value is primarily to provide a human-readable value.	Yes	
dateval	The value of the **gbauthdate** parameter expressed as the number of seconds since 1 January 1970 00:00 UTC. The Adobe GBLink service generates this value. The fulfillment server compares this date with the date on which it receives the order, and it rejects the order if the difference exceeds the order expiration period recorded in the store's entry in the eBook database	Yes	
gblver	The version number of the Adobe GBLink service. This value is generated by the GBLink component.	Yes	
auth	A digital signature generated by the Adobe GBLink service.	Yes	

Optional Parameter	Value	Content Server 4	Comment
custid	A string, in any form, identifying the customer to the fulfillment server.	No	
protocol	A string identifying the packaging and delivery protocol for the eBooks. Possible values are **ebx**	No	

	and **pdfm**. If this parameter is absent, the fulfillment server uses the EBX protocol.		
site	The key name, as recorded in the site's entry in the eBook database that identifies the download site where the customer's eBook reader obtains the book content. If this parameter is absent, the fulfillment server uses the default download site recorded in the store's entry in the eBook database	No	
price	The purchase price of the book. The value must be an integer or decimal number without a currency symbol.	No	
rights	**$**time**$$**copy**$$**print**$** Or $default$ A concatenation of specifications for usage time, copy rights, and print rights. Each of **$***time***$**, **$***copy***$**, and **$***print***$** can appear zero or one time and can appear in any order. The value **$default$** means to use the usage time, copy rights, and print rights recorded for the book in the eBook database Note that if no rights are specified then the default is that there are no additional restrictions beyond what is specified in the default rights specified when the resource was packaged.	Yes	Note that Content Server 4 does NOT support multiple books per URL

11.4 Rights narrowing

When a book is initially packaged by the Content Server 4 operator, it is given a set of permissions. Those permissions comprise the default permissions for the book. However, the distributor can specify additional permissions, but these permissions can only NARROW the actual permissions. In other words, the additional permissions cannot broaden the user's rights to the resource, but only further restrict them.

This narrowing of the rights is eventually performed by the Content Server 4 fulfillment service when the fulfillment token is processed during the fulfillment process.

11.5 URL Generation

When GBLink is used to sign a query string, several actions need to occur, including

- Adding gbauthdate, dateval, and the gblver parameters to the query string.
- Certain security steps need to be taken to ensure that the URL is securely signed
- The parameters in the string need to be URL-encoded

All three of these actions are performed automatically by GBLink – the Distributor does not need to take any action to achieve these.

11.6 Specification for the rights parameter

The optional rights parameter is: rights=*concatenated rights expressions*. If the rights parameter is not specified, the default set of rights for the distributor for that resource is used. The following sections detail the rights that are supported by Content Server 4.

11.6.1 Loan Relative Time

The syntax is:

$lrt#tttt$

Where 'tttt' is the number of seconds of ownership. The loan will expire 'tttt' seconds after the license is created. This will be converted into an absolute time when the License Token is created.

11.6.2 Loan Absolute Time

The syntax is:

$lat#tttt$

where 'tttt' (Expiration date) is the number of seconds since midnight, January 1, 1970 UTC. The Loan will expire at this Exact UTC time.

11.6.3 Copy permission

The syntax is

$cpy#nnn#tttt$

Copy 'nnn' selections to clipboard every 'tttt' seconds, where 'nnn' is the event count (number of clipboard operations) and 'tttt' is the number of seconds for the interval.

- To revoke the permission, simply specify 0 instead of nnn#tttt.
- To specify the max number over the lifetime of ownership, set the 'tttt' time interval to the lower-case string 'ever'
- To grant unlimited copying, DO NOT SEND the $cpy# permission, the max available as packaged will be granted
- Negative values for nnn or tttt are not allowed!

Examples:

Value	Resulting Permission
$cpy#15#86400$	Copy 15 selections to clipboard every 1 day (86400 secs)
$cpy#30#3024000$	Copy 30 selections to clipboard every 35 days (3024000 secs)
$cpy#17#0$	NOT ALLOWED; error
$cpy#0$	Copy permission is revoked (not available)
$cpy#0$	Interval ignored; same as $cpy#0$; Copy permission is revoked (not available)
$cpy#0#86400$	Interval ignored; same as $cpy#0$; Copy permission is revoked (not available)
$cpy#25$	NOT ALLOWED; error
$cpy#200#ever$	Copy 200 selections to clipboard ever (over the lifetime of ownership)
( )	DO NOT SEND $cpy# to grant unlimited (max available) copy permissions

11.6.4 Print permission

The syntax is

$prn#nnn#tttt$

Print 'nnn' pages every 'tttt' seconds where 'nnn' is the event count (number of pages permitted to print) and 'tttt' is the number of seconds for the interval.

- To revoke the permission, simply specify 0 instead of nnn#tttt.
- To specify the max number over the lifetime of ownership, set the 'tttt' time interval to the lower-case string 'ever'
- To grant unlimited printing, DO NOT SEND the $prn# permission, the max available as packaged will be granted
- Negative values for nnn or tttt are not allowed!

Examples:

Value	Resulting Permission
$prn#15#86400$	Print 15 pages every 1 day (86400 secs)
$prn#30#3024000$	Print 30 pages every 35 days (3024000 secs)
$prn#17#0$	NOT ALLOWED; error
$prn#0$	Print permission is revoked (not available)
$prn#0#86400$	Interval ignored; same as $prn#0$; Print permission is revoked (not available)
$prn#25$	NOT ALLOWED; error
$prn#200#ever$	Print 200 pages ever (over the lifetime of ownership)
()	DO NOT SEND $prn# to grant unlimited (max available) print permissions

12 Packaging Books with Content Server 4

12.1 Introduction

The packaging tool is a simple utility written in Java™ that is supplied with the Content Server 4 deliverables. It is a fully functioning tool that can be used to package books in bulk. It is also provided as a demonstration and helper code to help aggregators in integrating the packaging step with their systems. For this reason the source code is provided as well as the compiled JAR file.

12.2 Packaging Tool Example

Content Server 4 is delivered with a sample packaging tool written in Java that uses the UploadTest class. The sources are included in the archive along with a pre-built jar. Customers are encouraged to study the code and are welcome to copy and/or modify the code as they see fit. Please read the copyright notice in the files however, which limits Adobe's responsibility with respect to the code.

This tool creates and sends a Package Request to a specified packaging server and receives the server's response. The Package Request is an XML-based communication that specifies information about the book being packaged, and includes the book bytes in Base64 encoding, as well as an HMAC signature based on a SharedSecret (see XMLUtil Class for more information on this) The structure of the Package Request is shown below (elements marked `<!-- OPTIONAL -->` are optional):

```
<package xmlns="http://ns.adobe.com/adept/>
    <metadata xmlns:dc="http://purl.org/dc/elements/1.1/"> <!— OPTIONAL -->
        <dc:title><!-- OPTIONAL --> Book Title</dc:title>
        <dc:creator><!-- OPTIONAL --> Book Creator</dc:creator>
        <dc:publisher><!-- OPTIONAL --> Book Publisher</dc:creator>
        <dc:format><!-- OPTIONAL --> Book mimetype</dc:format>
    </metadata>

    <permissions><!-- OPTIONAL -->
        <display><!—OPTIONAL unless excerpt or print specified --> rights
```

```
elements</display>
        <excerpt><!-- OPTIONAL --> rights elements</excerpt>
        <print><!-- OPTIONAL --> rights elements</print>
    </permissions>
    <data> Base64-encoded book bytes </data>

    <thumbnailData><!-- OPTIONAL --> Base64-encoded thumbnail bytes </thumbnailData>

    <expiration> W3CDTF expiration </expiration>

    <nonce> Base64-encoded nonce </nonce>

    <hmac> Base64-encoded HMAC </hmac>
</package>
```

When you run this tool, the first parameter MUST be the URL of the Packaging server to send the Packaging Request to (EX: "http://ugra.corp.adobe.com/packaging/Package") The second parameter MUST be the file name of the target file to package or a directory name (EX: "/Users/piotrk/documents/testPDF.pdf") If the tool is supplied with a directory name, it will attempt to package every ".pdf" and ".epub" file contained in the directory. It does not matter whether or not you include a single slash (/) at the end of the directory name.

Important: When you invoke the tool, be sure to tell Java to give the process enough heap size. At least 512 MB is recommended and more if the books are large and/or complex. Heap size is specified with the command

```
java –Xmx512M
```

This tool accepts a number of parameter flags that allow the addition of DC metadata, permissions, and a thumbnail image into the Package Request. The order of the parameter flags can be arbitrary. Below is a list of accepted flags:

Metadata:

title	the next argument contains the DEFAULT dc:title value
creator	creator = the next argument contains the DEFAULT dc:creator value
publisher	the next argument contains the DEFAULT dc:publisher value
format	the next argument contains the DEFAULT dc:format value

Miscellaneous Flags:

png	looks for fileNameNoExt.png to upload as a thumbnail
jpeg	looks for fileNameNoExt.jpeg to upload as a thumbnail
jpg	looks for fileNameNoExt.jpg to upload as a thumbnail
gif	looks for fileNameNoExt.gif to upload as a thumbnail
xml	looks for fileNameNoExt.xml to use as XML source
verbose	displays content of package request and detailed server response
version	displays tool's version number
?	displays the list of accepted command-line flags

When the DC parameter flags are used, the parameter immediately following the flag is the value that is assigned as a default for that DC metadata element. (e.g. if you run the tool with "-format application/pdf", the default value of the DC metadata element <format> will be set to application/pdf)

When one of the thumbnail flags is used (-png, -jpeg, -jpg, or -gif), the tool will look for a thumbnail image with the corresponding extension in the same directory as the target file to be packaged. (e.g. if you run the tool with -png and the file name foo.epub, the tool will look for foo.png to package as a thumbnail for the .epub book). If the tool is supplied with a directory, it will look for a thumbnail image with the corresponding extension for each file in the directory.

The tool also accepts multiple thumbnail extension flags. This was included to allow the tool to package a directory of files that have thumbnails with different extensions. The tool will look for thumbnails that have the same base name as the file and then checks to see if there exists a thumbnail image file with one of the extensions provided by the command-line flags used.

When the -verbose flag is used, the tool will display detailed content to the console. The tool will display the path of the file being packaged, the path of the thumbnail image used (if present), the path of the XML source used (if present), the full Package Request, and the full server response. WARNING: The Package Request includes the Base64-encoded book bytes, and using -verbose will often overflow the default console buffer

When the -xml flag is used, the tool will look for an XML file with the .xml extension in the same directory as the target file to be packaged (e.g. if you run the tool with -xml and the file name foo.epub, the tool will look for foo.xml to use as XML source) The XML source file can contain specific DC metadata and permissions that will be used for the book. If an XML source file is found, the DC metadata within will be used in place of any default values set by the DC parameter flags. Including an XML source file is the only way to set permissions using this tool.

The XML source file MUST mimic the structure of the package request. Any elements other than <metadata> and <permissions> will be ignored. This tool will not accept XML that is not well-formed. Below is some sample content for an XML source file.

```
<package xmlns="http://ns.adobe.com/adept">
    <metadata xmlns:dc="http://purl.org/dc/elements/1.1/">
        <dc:title>XML Config Test</dc:title>
        <dc:creator>Piotr Kula</dc:creator>
        <dc:publisher>Adobe Systems Inc</dc:publisher>
        <dc:format>application/pdf</dc:format>
    </metadata>

    <permissions>
        <display>
            <device/>
            <until>2008-06-11T10:10:49-07:00</until>
        </display>
        <excerpt>
            <until>2008-06-11T10:10:49-07:00</until>
        </excerpt>
        <print>
            <count initial="10" max="20" incrementInterval="3600"/>
            <maxResolution>300</maxResolution>
        </print>
    </permissions>
</package>
```

Note: More detailed information on the packaging web-service (and hence the tool as well) can be found in section 7.3.

13 Fulfillment in Content Server 4

13.1 The Fulfillment Process

The fulfillment process is handled by Digital Editions on the client and the Fulfllment service in Content Server 4. So the process is largely opaque to the typical Content Server 4 operator. However, it is worthwhile to briefly cover the details of the process.

1. Distributor Generates the Signed URL

   The distributor creates the signed URL using the methods detailed in section 10.3 (Signed Link Usage). This URL is digitally signed using an HMAC so that the Content Server 4 fulfillment service can be sure that a valid distributor was the originator of the URL.

2. Distributor creates thank you page with embedded URL

Upon receipt of the end-user's authentication and/or payment, the distributor responds with the thank you page with the GBLink URL embedded in it. Note that the URL could also be embedded in an email or other form of internet delivery. The user then triggers the next step in the process by clicking on the URL, which causes the request to be sent to the Content Server 4 fulfillment service which is the target of the URL.

3. Content Server 4 Fulfillment service generates the fulfillment token

   The Content Server 4 fulfillment service receives the URL request. It checks that the HMAC matches the expected signature for that distributor. If the signature does not match the request is rejected. If it matches, the URL is parsed to obtain the requested book (resource) as well as any additional permissions. Each book has implicit permissions that were set when it was packaged, but distributors may specify additional permissions that further restrict the usage of the book. This process is known as "narrowing the permissions". Distributors can only narrow the permissions, they cannot expand them.

   Once the URL has been parsed, the fulfillment service constructs the fulfillment token for this transaction. The fulfillment token is constructed from the information in the URL and the Content Server 4's own information. The fulfillment token is then sent as a response to the URL request down to the browser as a small XML with a mime type for which ADE is registered to handle (*application/vnd.adobe.adept+xml*)

   Note that the "transaction" refers to a single fulfillment token issued, which can potentially contain multiple resources.

4. ADE saves the fulfillment token on the client machine

   The fulfillment token is saved locally so that in case the fulfillment process fails, the process can be restarted without having to re-fetch it from the server.

5. ADE checks if the user has used this Content Server 4 operator before

   If the user has used this operator before, then both client and server already have exchanged certificates, so that step can be skipped. If not, the certificates are exchanged and the operator's URL is added to the list.

6. ADE sends fulfillment request to the Content Server 4 Operator

   When ADE receives the fulfillment token, it creates another XML message (fulfillment request), that includes user id and the complete fulfillment token XML, signs it with the user credentials and sends it to Content Server 4 server (whose URL is extracted from the fulfillment token).

7. Content Server 4 server processes the fulfillment request

   When the Content Server 4 server receives the purchase request, it validates it by checking:

   - user signature on the fulfillment request is valid
   - distributor signature on the fulfillment token is valid
   - distributor has a right to distribute the resource(s) referenced in the fulfillment token
   - fulfillment token expiration time is not too far in the future (does not exceed the maximum request expiration interval)

   Also, the Content Server 4 server determines if this request is a new fulfillment (and a new license needs to be obtained) or a re-download. Request is a new fulfillment if it is not expired and the transaction id was not used by this distributor recently (within the last maximum request interval for Content Server 4 server). If this is a re-download request, Content Server 4 simply finds the existing license for the user and resource(s) and sends down the license using only that as an input (and the next two steps are skipped).

8. If new request, fulfillment service generates a new license

   If this is an initial fulfillment request, a new license is created for each resource in the fulfillment token. The license is generated by calling the generateLicense() web-service with the appropriate information derived from the user info, etc.

9. Content Server 4 server submits the license token to Adobe's license signing service

The license token, which is the core of Content Server 4 license, is digitally signed by the Content Server 4 operator and sent to Adobe signing server. The signing request can also contain an unsigned loanToken.

10. Adobe's license signing service signs and returns the license token

    Adobe signing server validates Content Server 4 operator's signature, removes it and signs the license token with the Adobe private license key. If a loanToken is present, it will be signed and returned as part of the same response.

11. Content Server 4 server sends the license to ADE

    The Content Server 4 saves the signed license in the database and then returns the complete license containing the signed license token to ADE.

12. ADE looks at the licenseURL in the licenseToken and downloads the certificate that corresponds to that licenseURL.

    This is done through LicenseServiceInfo request to fulfillment service.

13. ADE contacts Media Server to get the encrypted resource

    ADE extracts the Media Server URL from the license. It then requests bookbytes from the Media Server.

14. Media server provides the encrypted resource to ADE

    This is a simple HTTP download.

15. ADE writes the license into the encrypted resource

16. ADE notifies Content Server 4 Operator and possibly the Distributor that fulfillment completed successfully

13.2 Why Limiting Retries of Links Isn't Needed Anymore

One aspect of Content Server 3 usage that led to increased support requirements by businesses operating Content Server 3 was the need to re-supply users with new links when their download failed. An Content Server 3 fulfillment can fail for many reasons (no compatible viewer installed, network failure, a bug in the software, etc.) To overcome that, a customer is typically given several attempts to download the book. The number of attempts in configurable and can change from store to store. Since every download is fulfilled independently (Content Server 3 does not keep track of all existing vouchers it has already produced), there has to be a cap on the number of attempts: otherwise a person could potentially buy a book and publish the download URL for everyone else to get a (seemingly legitimate) copy. Note that this would only be possible if the Aggregator had set the Content Server 3 fulfillment service to allow multiple downloads per URL. Even today a malicious user can abuse multiple download attempts to produce a copy of the book for his friends. On the other hand, if a legitimate user encounters problems and reaches that cap he has to call customer support to reactivate that link.

Obviously Content Server 4 fulfillment can fail as well, but several significant improvements have been made to the Content Server 4 protocols that make failure recovery much more robust. Fundamental to these changes is that the fulfillment token that is downloaded to the client (ADE) is digitally signed before being delivered. Since the token is therefore unique and cannot be altered (without making it invalid), there is no need to track that a transaction has been started but not completed. The database state is unchanged. In fact, as far as Content Server 4 is concerned, there is no transaction until the fulfillment token is sent to the Content Server 4 server by the client (ADE). Once the fulfillment token reaches the Content Server 4 server, the license is generated and saved in the Content Server 4 database. From the Content Server 4 point of view, the transaction is now complete - in one single step. Then the resulting license is downloaded to the user.

This makes the logic simple and robust - there are no incomplete transactions in Content Server 4. The transaction is either not known to Content Server 4 or it is complete. If the transaction is not known to the server, the download link stays active until it expires, but there is no sensitivity to the number of attempts. Once the transaction is complete, the license is stored in the database. If the user uses the link again, the exact same license will be fetched from the database and downloaded again. This process can be repeated an arbitrary number of times because the user's ID is already burned into the license during the original fulfillment. If a user shares the download link with another person after the

purchase, the other user won't be able to utilize it because the Content Server 4 server will detect that the second user's ID does not match the ID in the requested license and will report an error. As a result of this approach, there is no need to cap the number of downloads so the whole problem with limiting the number of downloads and tracking incomplete transactions just goes away.

13.3 The License

The Content Server 4 License is XML file which enables user to open Content Server 4-protected file.

Content Server 4 License contains the following information:

- license token, which contains
    - resource id
    - user id
    - resource key, encrypted with the user's public license key
    - permissions
    - digital signature
- license server information
    - license server URL
    - license server certificate

In EPUB files, the Content Server 4 license is stored in the package in this location:

META-INF/rights.xml

In PDF files, the Content Server 4 license is stored in the EncryptionDict under the /ADEPT_LICENSE key. It is gzip-compressed and then base64-encoded and stored as a string.

Here is a sample Content Server 4 license:

```
<?xml version="1.0"?>
<rights xmlns="http://ns.adobe.com/adept">
    <licenseToken>
        <resource>urn:uuid:8e91a04d-ed12-4c1f-ac70-2047016bc371</resource>
        <user>urn:uuid:16f750dd-3e70-426d-873d-e291ae43a317</user>
        <licenseURL>http://ugra.corp.adobe.com/acs4</licenseURL>
        <operatorURLhttp://ugra.corp.adobe.com/fulfillment</operatorURL>
        <encryptedKey>eeR+rMD7...G+4bWQG+s=</encryptedKey>
        <permissions>
            <display/>
            <excerpt/>
            <print/>
        </permissions>
        <signature>z849Ra...xN5z7uXzf8mBg=</signature>
    </licenseToken>
    <licenseServiceInfo xmlns="http://ns.adobe.com/adept">
        <licenseURL>http://ugra.corp.adobe.com/acs4</licenseURL>
        <certificate>MIIDyzCC...T0mPxnvfR</certificate>
    </licenseServiceInfo>
</rights>
```

Note that the license as a whole is digitally signed, so even though it is present in the document and can be relatively easily extracted and/or removed, the Digital Editions client software will not open a document which does not have a valid, properly signed license – or if the license has been modified or otherwise tampered with.

Further, it is not possible to open the document without the license not only in ADE, but in any EPUB or PDF renderers because it is the license that contains the encryption key. Also, machine activation info is needed, because the encryption key is encrypted with the user's private key which is transferred to the machine as a part of the activation. So without the properly encrypted and signed license, the document cannot be opened by ADE or any other software.

13.4 The Fulfillment Token

The fulfillment token is returned to Digital Editions by the fulfillment service after it has processed the signed URL it receives from the client. It contains:

- distributor id
- unique transaction id (for that distributor)
- the license token
- resource id(s)
- (optional) resource permission set
- Content Server 4 operator URL
- request expiration time
- distributor's signature (PKI or HMAC)

It also may contain a loan token (if it is not a purchase but a loan). It may also contain a notification element if the Distributor has requested (in the signed URL) to be notified when the transaction is complete.

The exact details of the syntax are not really relevant here as the generation of the token and its consumption by Digital Editions are opaque to the Distributor and Content Server 4 operator.

The details of the fulfillment token syntax are covered in sections 1.8 and 4 of the Content Server 4 Technical Reference.

13.5 The Loan Token

The Loan Token is an important element of the Content Server 4 infrastructure. The fulfillment info returned to the client with the signed license also contains the loan information if the document is a loaned item. This loan information is processed by the client and the loan token object on the user's machine is updated.

The loan object is small file that contains the list of books that the user has borrowed. Each loan token contains information about the book, expiration time, etc. The loan token file is also digitally signed so it cannot be tampered with. The client always checks the loan token when it opens the book to verify that it has not expired, etc. Note that this loan token is also transferred to a tethered mobile device (such as the Sony Reader) so that the loan permissions can be enforced there as well.

14 Using the Operator Client

14.1 Purpose

The Operator Client is a tool for managing your account on the license signing server. It provides access to getting logs of any signing operations from your fulfillment servers as well as tools for managing the certificates for any URL on which you have fulfillment server.

14.2 Launching

The operatorClient may be launched in most operating systems by just double clicking "operatorClient.jar"

NOTE: In order to handle reports with thousands of items, the memory size must be increased. In order to do so, run the following command:

```
java –jar –Xmx1024m <path>/operatorClient.jar
```

You may wish to write a script or batch file to automatically run this command.

14.3 Login Tab

When first launching the product, you will see the Login tab.

Make sure that the license signing URL is the URL which was provided for you. Please note that this URL should point to “licenseadmin” instead of “licensesign” which is used in the fulfillment configuration files. Once this is changed, clicking “Save” will save the server locations between runs of the application.

Type in your license signing username and password and click “Login”.

14.4 Transaction Reports Tab

The transaction reports tab allows you to retrieve various types of reports of signing transactions. The three types of reports are:

1. Summary
   Get Summary will return a text summary of the number of signing transactions sorted by domain name and type. This is the way to easily see the number of buy and loan transactions for each domain as well as the total number of transactions for this time period.
2. XML listing:
   - This will return a listing of all transactions matching the date range in a XML format.
   - A maximum of 50,000 transactions may be requested at a time.
   - XML format:

```
<transaction>
    <ref> EUID </ref>
    <fulfillment> EUID </fulfillment>
    <operatorURL> String - URI </operatorURL>
    <transactionTime> dateTime </transactionTime>
    <transactionType> int </transactionType>
    <user> Eden 2 GUID </user>
    <resource> UUID </resource>
    <oldPerm> int </oldPerm>
    <newPerm> int </newPerm>
</transaction>
```

3. CSV listing:
   This will return a listing of all transactions matching the date range in a comma separated value format.
   - A maximum of 50,000 transactions may be requested at one time.
   - The columns provided are in the following order:
     ref, fulfillment, operatorURL, operatorTime, user, resource, oldPerm, newPerm

14.4.1 Date ranges:

In order to set a date range, just input the new date and time in the date spinner. In order to leave a date open ended (for instance, getting all transactions from yesterday forward) simply uncheck the date which should be open ended.

14.4.2 Saving reports:

By default all reports are printed to the text area supplied. In order to save to a file, simply run the report then click the “Save to File” button on the bottom of the screen in order to save the last run report to a file.

14.4.3 Blank response:

If you request a report with a large number of transactions, sometimes the result field stays blank. This is usually caused by the program not having enough memory to handle the results. In order to test if this is the case, run the application from the command line. If you get an OutOfMemory error printed out to the console, restart the application with a higher memory amount as described in the Launching section.

14.5 Operator URLs Tab

This tab will show a list of URLs that this user has certificates for on the server and allow this list to be modified. Each URL listed will have the expiration date of the certificate for that URL. A URL may have multiple certificates associated with it, but this should only be used for transitioning between a certificate which is close to expiring to a new certificate. Once the new certificate is up and running the old certificate should be revoked.

Operations:

1. Revoke
   Revoke will revoke the certificate for the selected domain. Revoking will prevent the certificate from being checked when checking validity of a certificate.
2. Get Cert
   The Get Cert command will download the certificate stored for the selected URL to a local file. This will be the same certificate that was returned from an Add operation
3. Add
   "Add" will create a certificate for the URL specified in the URL text field using a given P10 file. This certificate will be saved on the server as well as being saved in the location specified in "Result file". This certificate can then be used to create a p12 file used by the fulfillment server as defined by Operator Enrollment in the User's Guide.
4. Update List
   Update List will simply repopulate the list with the current data from the server. Any operations will automatically update the list, so this is only necessary if the list has been modified from a different source while this program is running.

14.6 Adding a URL

If you wish to add a fulfillment server, the following steps will set up the license signing server to handle this new URL:

1. Start the application and Login
2. Go to the "Operator URLs" tab
3. Enter the new URL in the "URL" field. Make sure to include the full path to the fulfillment server. For example "http://www.yourdomain.com/fulfillment"
4. In the P10 file field, select a P10 file you created (See Operator Enrollment in the User's Guide for details)
5. The Result File field will default to the file being in the same directory with the same name but a .cer extension. If you want a different location, change it now.
6. Click the "Add" button
7. Verify that the new URL has been added to the list.
8. Follow the steps in Operator Enrollment in the User's Guide to take this certificate and make it a p12 file for use on your fulfillment server.

   (If you lose this certificate file somehow you can always come back, select the appropriate URL from the list and click "Get Cert" in order to get another copy of this certificate file.)

15 Admin Console

15.1 Overview

The Admin Console is a Flex based application for viewing and editing the current state and the interrelation of the persistence objects. The primary purpose of the Admin console is to provide operators with a technical reference tool to aid in the initial implementation of Content Server 4 by serving as an interactive illustration of key features and the correlating API requests and responses. While some operators may find the Admin Console occasionally useful for specific tasks once Content Server 4 is fully implemented within their environment, it is not intended to serve as a production workflow tool. The Admin Console is in early development stages and is not fully implemented. In its current state it has three main views: Inventory, Distributors, and Licenses. All views include an API inspector at the bottom, which displays the requests and responses to the persistence objects associated with each user gesture. Future versions of the Admin Console may include (among other improvements) an import feature, to allow users to add new resource items via the Admin Console, and a configuration feature, to make Console setup easier. The source code of the Admin Console is also included as sample code within the archive zip file.

15.2 Setup

The Admin Console application is included in the admin.war file. To launch the admin console, open *http://<war file location> /admin/console/* -where "<war file location>" is the URL of your war file deployment. Note that the client machine must have Flash Player 9 or better installed to run the Admin Console application.

15.3 Logging in

Each time Admin Console is launched a login dialog will appear. The dialog contains an editable field containing the path to the admin server and a password field.

15.3.1 Login after initial setup with default password

If the console is being launched for the first time, the default password is the one you were sent in the welcome email from Adobe and should be use to log-in. In this case, upon successful log-in, the user will be prompted to change the password. When the new password is submitted, a new Shared Secret value will be generated by the Admin Console and stored in the database.

15.3.2 If Login Password is forgotten

If the password needs to be reset due to a forgotten password the shared secret value for the default distributor in the distributor table in the database (Distributor ID is "1") can be manually changed back to the initial value (that you were sent in the welcome email from Adobe) and that value may be again used when logging in to the admin console.

15.3.3 Changing the Password

Click the "change password" menu item within the "More" menu located in the upper right corner of the Admin Console UI. When a new password is submitted, the admin console will generate a new Shared Secret value for the default distributor and store it in the distributor table in the database.

15.4 Using Inventory View

15.4.1 Overview:

The inventory view provides tools for viewing resource (book) data and viewing/editing distribution rights data. Note that within the Admin Console UI, resources are referred to simply as "items". Also, the term "inventory" is used in the Admin Console to refer to a set of items. This is usually used to refer to a set of items that all have associated distribution rights data referencing a common distributor (a "Distributor Inventory"). It is also used to refer to the list of all items within an operator's fulfillment database (the "Operator Inventory").

15.4.2 View the entire operator inventory:

- Click the "All Items" label found near the top of the left pane

15.4.3 View a distributor's inventory

- Click the distributor's name label in the left pane to view the list of items with associated distribution rights applying to that distributor

15.4.4 Add an item to a distributor's inventory

- Select item(s) within the center pane (when viewing either the operator's inventory or another distributor's inventory)
- Drag the items onto the name label of the target distributor
- Note that this will automatically create a new distribution rights type labeled "default" associated with the item and referencing the target distributor's UID

15.4.5 Remove an item from a distributor's inventory

- Select item(s) within the center pane (when viewing a distributor inventory)
- Drag the item(s) onto the trash can icon in the footer of the pane

15.4.6 Deleting an item from operator Inventory

- Select item(s) within the center pane (when viewing the Operator Inventory)
- Drag the item(s) onto the trash can icon in the footer of the pane
- Note that this will also remove the item from any distributor inventories that it was in

15.4.7 Viewing/Editing base permissions for an item

- Select the target item in the center pane
- The base permissions details will appear in the "Base Permissions" inspector panel found in the right pane
- Base permission can be edited via controls in the inspector panel by clicking on the "edit" menu item in the contextual menu located next to each permission class label (Read, Copy, Print)
- It is important to note that edits to base permissions are independent of the inventory currently being viewed, and apply to the item across all inventories –including the operator inventory, and are not specific to a particular distributor
- For a more detailed description of the UI workflows around editing permissions, refer to "Editing a permission within the Distribution Rights inspector" below as the interactions are very similar

15.4.8 View an item's distribution rights

- Select the item in the center pane when viewing a distributor inventory

- The distribution rights details will appear in the "Distribution Rights" inspector panel in the right pane
- This pane is further sub-divided into sections that represent "Distribution Rights Types" – named "sets" of permissions
- Each item will have at least one Distribution Rights Type, and can have up to ten. Each type has a name label, and a display of the permissions associated with that type
- Note that an item's distribution rights are specific to each Distributor

15.4.9 About Distribution Rights Types

The display of Distribution Rights Types details within the Distribution Rights Inspector panel is visually separated into discreet permissions. At a minimum this consists of one (each) Read, Print, Copy, and Loan permissions. However, it is possible to have multiple Read, Print, or Copy permissions in a single distribution rights type.

For example, a Distribution Rights Type could contain one print permission that allows for a small page count of hi-res prints, and another print permission that allows for a larger page count of low-res prints.

Each permission can have multiple different parameters such as expiration, accruals, device type constraints, etc. This offers extremely granular and flexible control over permissions.

It is important to note that it is possible to create a permission that is less restrictive than the base permissions for the item, or that is less restrictive than another permission within the same Rights Type. However, this less restrictive permission will essentially be "ignored" when the item is fulfilled, as the most restrictive permission will apply.

An example of how this feature might be used in the Admin Console is to create one Rights Type labeled "Purchase" with generous printing and copying permission and in with a read right that does not expire, and another Right Type labeled "Preview" with restrictive printing and copying permissions and the read right expires 1 hours after the item is fulfilled. The distributor indicates the Rights Type to be applied at the time of the fulfillment request.

15.4.10 Adding a new Type to an item's Distribution Rights

- Select the target item in the center pane.
- Click the "Add Rights Type" button found in the header bar of the Distribution Rights Inspector panel (the square button with a "+" icon)
- Enter the desired name of the new type and click on "Save"

15.4.11 Editing a permission within the Distribution Rights Inspector

- Mouse over a permission details string (e.g. the description of a Print permission)
- Click the contextual menu button that appears
- Click the "edit" menu item.
- Modify parameters and settings in the dialog that appears
- Click "Update"
- Note that if you have multiple Read, Copy, or Print permission within a single Rights Type, that selecting the "not allowed" radio button when editing a single permission will cause any other permission of the same class to be deleted. E.g. if you have multiple print permissions in a given Rights Type, and you edit one to the "printing not allowed" setting, then the other Print permissions will be deleted once the edit is saved (as they would no longer apply due to the "complete" restriction.)

15.4.12 Add a new permission to a Distribution Rights Type

- Click the “add permission” button (a square button with a “+” icon) next to the name label of the desired permission class (e.g. Read, Copy, Print)
- Modify parameters and settings in the dialog that appears
- Click “Update”

15.4.13 Change the name label of a Distribution Rights Type

- Click the “view details dialog” button (looks like an “i” in a circle) found next to the target Distribution Rights Type name label in the Distribution Rights Inspector Panel
- Edit the name value in the editable field provided
- Click “Update”

15.5 Using Licenses View

15.5.1 Overview

The Licenses view of the Admin Console provides tools for viewing a list of licenses issued by an operator, viewing license and user details, and deleting users or licenses.

15.5.2 View all licenses

- Click the “All Licenses” label found at the top of the left pane

15.5.3 View licenses associated with a specific User ID

- Click on the desired User ID in the left pane, or use the search control at the top of the center pane when viewing all licenses

15.5.4 Deleting a License

- Select a license list item in the center pane
- Drag it onto the trash can icon found in the footer of the pane

15.5.5 Deleting a User

- Select an item associated with a target user from the list in the center pane
- Click on the “delete user” button found in the User Inspector panel in the right pane (looks like a trash can icon)
- Note that you cannot delete a user that has associated licenses. You must first delete the licenses to delete the user.

15.6 Using Distributors View

15.6.1 Overview

The Distributors View provides tools for viewing the list of distributors in an operator’s fulfillment database, viewing and editing distributor details, and exporting Sample Stores.

15.6.2 Add a New Distributor

- Click the “Add Distributor” button in the footer of the center pane (looks like an orange square button with a “+” icon)
- Fill in the fields
- Click “Save”

15.6.3 Edit Distributor Details

- Select the target distributor list item in the center pane
- Edit the values in the “Distributor Info” inspector panel in the right pane
- Click Update”

15.6.4 Change a Distributors Shared Secret

- Select the target distributor list item in the center pane
- Click the “Distributor Details Dialog” button in the header of the Distributor Info Inspector (looks like an “i” in a circle)
- Click the “new shared secret” link in the dialog
- Close the dialog

15.7 Setting Up A Sample Store

Creating a Sample Store php website for a distributor requires using the Admin Console to create the configuration (config.xml) and the content list (books.xml).

15.7.1 Export Store Files:

- Go to the “Distributors” view
- Click a distributor in the list in the center pane to select it
- Click the “Export Store” button (this creates books.xml for the selected distributor)
- The exported file will include all items included in the distributor’s inventory.
    - To see a list of the items that will be included, go to Inventory view and click on the distributor’s name in the left pane.
- Click the “Export Config” button (this creates config.xml)
- Place books.xml and config.xml alongside the rest of the store .php files.

16 Sample Store

16.1 Overview

The sample store that is supplied with the Content Server 4 release is intended to facilitate Aggregators and Distributors becoming familiar with using the Content Server 4 fulfillment services. It is clearly not intended as production code, though you are free to use it as you see fit.

The store consists of 3 pages:

- Catalog (catalog.php),
- Details (details.php)
- Thank You (thankyou.php).

All pages have the same shared header (header.php). The header adds the Adobe logo and search functionality.

The Details and ThankYou pages share the same footer (footer.php). The Footer adds "Return to catalog" functionality. Upon return the catalog will be positioned of the last selected book.

Other files used in store are:

- config.php - includes constants and variable definition.
- functions.php - include common functions, used in store
- xmlinit.php - xml initialization lines, common for all pages
- books.css - store's cascade style sheet, used by all pages
- books.xml - actual list of books, included in store. This file can have a different name as long as it is so defined in config.php. File is generated by Content Server 4 server export
- Config.xml - store configuration, includes the shared secret, store ID, server URL and some other settings. This file can have a different name as long as it is so defined in config.php.
- index.htm - html file, contains the redirection to catalog.php
- index.php - php file, contain redirection to catalog.php
- subfolder /images/ - contains adobe logo and "no image" images.

16.2 Installation

The store does not require any installation other than to copy it to any subfolder within the web site folder. If the web site's root domain is *www.domain.com* and store's files were copied to a */bookstore/* subfolder, then final store URL will be *www.domain.com/bookstore/*

Known installation issues are described in the known issues section (13.7).

16.3 Store Use/Workflow

The typical store use case is:

- Open catalog page. Choose a book and click on the purchase/details shortcut.
- Review the book's details, choose some permissions (if needed), then click on the "purchase" or "borrow" button
- Review the order information and resulting permissions, then click on the "download" link to start book download
- ADE will detect the downloaded ACMS file and complete the fulfillment, including downloading the bookbytes, writing the license into the document and opening the document.

16.4 Search usage

The store supports a simple single line search, where the search key can be one word, several words or regular expression. Upon receiving the search request, the store's script will try to validate the search key as a regular expression. If it is not a valid regular expression, then a regular word search will be performed. The phrase will be parsed into separate words (space, comma and semicolon are valid delimiters).

The search the searches the book's Title, Description, Author and Book ID fields for the specified keywords or regular expression. If you entered more than one word, then search will apply every word to every field and shows all matches.

16.5 Details page

On the Details page you may examine and/or modify the book's permissions. The permissions shown are drawn from the store's *books.xml* file. For interpretation of the permissions, please see Section NN, Digital Rights Management in ACS 4.

Bear in mind that the permissions set in the store can ONLY serve to "narrow" (further restrict) the permissions for the book. The permissions cannot be weakened.

Once you have chosen a set of permissions, choose "Purchase" or "Borrow". This will take you to the "Thank You Page".

16.6 Thank you page

The "Thank You" page shows the results of your permissions choices for the specified book. Once you have verified that the choices are what you wanted, click on the "Download" link to start the download process.

16.7 Known Issues

16.7.1 Missing or Outdated MSMXL Package

In some cases, you Windows installation might not have the latest MSXML installed, which will cause the script will fail on the line:

```
$dom = new DOMDocument("1.0","UTF-8");
```

In order to resolve this, please install latest MSXML from

http://www.microsoft.com/downloads/details.aspx?familyid=3144B72B-B4F2-46DA-B4B6-C5D7485F2B42&displaylang=en

Note that this URL is correct as of June 2008 but may change over time.

16.7.2 Access denied issue

You may encounter an error message such as

```
msxml4.dll (0x80070005) Access is denied.
```

This error indicates that the process in which the script is running has insufficient access rights. You need to upgrade the access rights on the machine. See the following Microsoft KB for additional information:

(http://support.microsoft.com/default.aspx?scid=kb;en-us;820882#6)

16.7.3 PHP Packages

You need to be using PHP 5, which has the domXML API package.

If you attempt to add the PHP extension package domXML to an earlier version of PHP, you may encounter conflicts and errors. Possible error messages include

```
Warning: domdocument::domdocument() expects at least 1 parameter, 0 given…
```

```
or

Fatal error: Call to undefined method domdocument::createElement() in…
```

You have to have the PHP Package php_http.dll (PECL pecl_http:1.0.0-1.5.5) installed.

You can find it here: http://pecl.php.net/package/pecl_http

17 Content Server 4 Web Services

17.1 Fulfillment

The fulfillment service provides the decryption key and permission operation for the resources. In general it is called by the client (ADE) and is run by Content Server 4 operators. In general, Aggregators do not need to access these web services as this is handled transparently by the client (Digital Editions) and the fulfillment service itself.

17.1.1 LicenseServiceInfo GET request

Get license server information (certificate and activation service URL).

No parameters are needed.

Server response:

```
<licenseServiceInfo xmlns="http://ns.adobe.com/adept">
    <operatorURL>license server URL</operatorURL>
    <licenseURL>license server URL</licenseURL>
    <licenseCertificate>base64-encoded license service certificate</licenseCertificate>
</licenseServiceInfo>
```

PKI certificate contains license server URL and is signed by Adobe private key.

17.1.2 Auth POST request

Provide user's certificate and public license key to the license server.

Only needed if the server that issued the certificate was different from the license server.

Request body:

```
<credentials xmlns="http://ns.adobe.com/adept">
    <user>userid</user>
    <certificate>base64-encoded PKI certificate</certificate>
    <licenseCertificate>base64-encoded certificate with public license
key</licenseCertificate>
</credentials>
```

Server reply:

```
<success xmlns="http://ns.adobe.com/adept"/>
```

17.1.3 Fulfill POST request

Fulfill the purchase: a fulfillment token is used to acquire the license for a resource.

Request body:

```
<fulfill xmlns="http://ns.adobe.com/adept">
    <fulfillmentToken>
        <!-- license token from the retailer -->
    </fulfillmentToken>
    <user>userid</user>
    <device>deviceid</device>
    <signature>base64-encoded user signature</signature>
</fulfill>
```

In addition, if the fulfillmentToken element contains serial element, user should be asked to enter the serial number of the resource (a number of alphanumeric characters given by the serial element length attribute). The result given by the user should be uppercased, stripped from non Latin (26 A-Z chars), non digit (0-9) characters, and included as the content of the serial element:

```
<fulfill xmlns="http://ns.adobe.com/adept">
    <fulfillmentToken>
        <!-- license token from the retailer -->
    </fulfillmentToken>
    <serial>serial number</serial>
    <user>userid</user>
    <device>deviceid</device>
    <signature>base64-encoded user signature</signature>
</fulfill>
```

Server reply:

```
<envelope xmlns="http://ns.adobe.com/adept">
    <fulfillment>
        <id>unique fulfillment id assigned by aggregator</id>
        <resourceInfo>
            <resource>resourceid</resource>
            <voucher>(optional) ACS3 voucher id for ACS3-packaged documents</voucher>
            <resourceItem>resource item id</resourceItem>
            <src>resource download URL</src>
            <downloadType>simple|auth</downloadType>
            <metadata xmlns:dc="http://purl.org/dc/elements/1.1/">
                <dc:title>resource title</dc:title>
                <dc:format>resource mime type</dc:format>
                ...
            </metadata>
            <licenseToken>
                <resource>resourceid</resource>
                <licenseURL>license server URL</licenseURL>
                <operatorURL>ACS4 operator URL</operatorURL>
                <encryptedKey>base64-encoded key, encrypted with user public license
key</encryptedKey>
                <user>userid</user>
                <permissions>
                    <!-- rights granted for this resource -->
                </permissions>
                <signature>base64-encoded license server signature</signature>
            </licenseToken>
        </resourceInfo>
        ...
        (list of resourceInfo blocks)
        <notify>
            <!-- optional -->
            ...
        </notify>
    </fulfillment>
    <loanToken>

        <!-- optional -->
        ...
    </loanToken>
</envelope>
```

If there is a problem with a particular resource on the list, an error element can be added as a child of a particular resourceInfo element.

loanToken element can be used to find out exactly which loans a valid as of the date indicated.

Loan id is a short string

Note that fulfillment response will be enveloped to include loan token and, optionally, store notification token.

Loan token syntax:

```
<loanToken>
    <time>timestamp</time>
    <user>userid</user>
    <licenseURL>license server URL</licenseURL>
    <operatorURL>license server URL</operatorURL>
    <loan>loan id</loan>
    ...
    (list of all loan ids which are not expired and not returned as of date on the
timestamp)
    ...
    <signature>base64-encoded license server signature</signature>
</loanToken>
```

17.2 ACS4 Fulfillment Notifications

17.2.1 Client Side

1. ADE receives notification requests from the fulfillment server as a part of response to either book fulfillment or loan return request.
2. There may be any number of requests per single operation (e.g. it may be one request to notify the Aggregator and one request to notify the Distributor).
3. The notification request XML format is the following:

```
<notify>
    <notifyURL> URL to be notified </notifyURL>
    <body> ... notification message ... </body>
</notify>
```

   - The <notifyURL> element should contain the full URL to the notification service.
   - The <body> element may contain any valid XML.
4. In case of fulfillment operation, ADE sends the notification(s) after the operation is completed, i.e. when license and book bytes are downloaded and license is written into the document.
5. In case of loan return, ADE sends notification as soon as the request is received.
6. The notification sent by ADE has the following format:

```
<notification>
    <user>user id</user>
    <device>device id</device>
    <body> ... requested body ... </body>
    <signature>base64-encoded user signature</signature>
</notification>
```

7. ADE does not require server to send any particular response to the notification. Notification is considered successfully delivered unless one of the following happens:
   - HTTP status code indicates an error (e.g. 404 - URL not found; e.g. 500 - internal server error);
   - HTTP status code indicates success, but the response is an XML with the root element named "error".

In case of error, the notification task will be saved. On the next ADE start user will be promted to resume pending operations and another attempt to deliver notifications will be performed if user agrees.

17.2.2 Server Side

1. ACS4 Fulfillment Server sends the following notification request body:

```
<body>
    <fulfillment>ACS4 fulfillment id</fulfillment>
    <user>user id</user>
    <transaction>distributor's transaction id</transaction>
    <fulfilled>true</fulfilled>
    <returned>false/true</returned>
    <hmac>base64-encoded hmac using distributor's shared secrect</hmac>
</body>
```

   - The value of <returned> element is `false` for the fulfillment and `true` for the loan return.
   - The <hmac> element is not included if there's no key shared between Aggregator and Distributor.

2. ACS4 Fulfillment Server has the "FulfillmentNotification" API/servlet, which processes ADE fulfillment notifications. When the notification is received the value of the "confirmed" field of the respective record in the "Fulfillment" table in database is changed from "F" to "T".
3. ACS4 Fulfillment server always sends notification request when fulfills the document. This request <notifyURL> element contains the URL of "FulfillmentNotification" servlet, described above.
4. If the "notifyURL" field of "Distributor" table is not null or empty, ACS4 Fulfillment Server sends the request to notify the Distributor.
5. If the "com.adobe.adept.fulfillment.customNotifyURL" property is set to a non-empty string representing a URL, ACS4 Fulfillment Server sends ADE the request to notify this URL both for fulfillment and loan return operations.

17.2.3 Processing custom notifications

As mentioned above ADE does not require server to send any particular response to the notification. Only a valid non-error response is sufficient.

17.3 Packaging

The packaging web-service provides the access that allows Aggregators to “package” their content, i.e. take unencrypted books and optional additional metadata, thumbnails and produce an encrypted book plus additional info necessary for deployment.

Note that the packaging web-service is the only way that the packaging services can be accessed. There are no corresponding Java objects that can be used to access the packaging services.

Packaging service takes the file and a number of additional pieces of information, encrypts the file then uploads it to the media server and updates the ResourceKeyData and ResourceItemData tables with the record about the new resource. It also gives a built-in distributor (UUID=00000000-0000-0000-0000-000000000001) distribution rights for that resource.

When the packaged resource (and, possibly, the thumbnail) is uploaded to the media server, an **upload location** is used. Normally the base upload location is defined at the server config file (a file name or ftp URL). That base location is combined with the file name that is either supplied with the packaging request (fileName element) or is derived from the resource id (by adding .pdf or .epub). However, upload location can changed altogether by supplying it in the request.

Once the resource is uploaded to the media server, it is available through the **download location**. Note that in general, there is no method to automatically derive download location from the upload location. Both have to be configured. Again, normally, download location is created from the base download location in the server config and the file name (supplied or automatically derived), but can be overridden in the request.

Here is the request:

```
<package xmlns="http://ns.adobe.com/adept">
    <resource>optional resource id</resource>
    <voucher>optional voucher idfor GBLink fulfillment</resource>
    <resourceItem>optional resource item index</resourceItem>
    <fileName>optional file name to use for this packaged resource</fileName>
    <location>optional upload location for packaged resource (file name or FTP
URL)</location>
    <src>optional download location for the packaged resource (HTTP URL)</src>
    <metadata>
       ... metadata description, the whole metadata element is optional ...
    </metadata>
    <permissions>
     ... permissions description, the whole permissions element optional ...
    </permissions>
    <!– Either data or dataPath element, not BOTH →
    <data>base64-encoded resource</data>
    <dataPath>path to the unencoded file to be packaged</dataPath>
    <thumbnailLocation>optional thumbnail upload location (file name or FTP
URL)</thumbnailLocation>
    <thumbnailData>base64-encoded thumbnail (optional)</thumbnailData>
    <expiratiion>request expiration time</expiration>
    <nonce>nonce</nonce>
    <hmac>HMAC based on the server password</hmc>
</package>
```

Element Name	Description
resource	Content Server 4 resource ID, will be assigned by the server if not supplied
voucher	Aggregator's resource identifier, this can be used to identify the resource in GBLink
resourceItem	Resource item index, 1 is default (only needed when multiple items exist per license)
fileName	When location or src is not supplied, this name is combined with the base upload and download base URLs to make resource-specific upload and download URL
location	Upload location (file name or URL) where the packaged resource will be written
dataPath	A path (relative to the packaging server) to a file in lieu of sending the file data directly in the message. The file should be in the clear, i.e. not base-64 encoded
pass	Specify the password used for the shared secret if the dataPath is specified. See note below.
src	Download URL for the resource which corresponds to the resource's upload location
metadata	Resource metadata - the data which is provided there will be written in the packaged document (if the document supports it, e.g. PDF does not support dc:language)
permissions	Base set of permissions for this document. Permissions cannot be elevated beyond this set by any distributor
data	resource bytes (PDF or EPUB), base-64 encoded
thumbnailLocation	upload location for the thumbnail
thumbnailData	base64-encoded thumbnail (GIF, PNG or JPEG)
expiration	request expiration time, an hour from the request time is recommended
nonce	a base64-encoded set of bytes; the only requirement is that a new nonce must be used with every request
hmac	message authentication code based on the server's password (sharedSecret of the built-in distributor, by default, the password you were sent by Adobe)

Note: If the -datapath flag is used, then the dataPath Mode is enabled for the Tool. dataPath mode is designed to make packaging large books simpler. Instead of packaging a book that is local to the machine by re-encoding it and passing

the data a base64-encoded blob, dataPath lets the user specify a path to a location on the packaging server where the file resides.

When dataPath Mode is engaged, the tool will no longer look for .epub and .pdf files to package, and instead will look for .xml files in the directory that it was passed. In order to use dataPath Mode, you must pass the tool a directory (not a file). The XML config files must include a <dataPath> element containing the local path to the book to be packaged (on the packaging server). This <dataPath> element will be used instead of the <data> element that is usually included in the request. A request may not have both data and dataPath elements. If dataPath Mode is not engaged, the Tool will ignore any dataPath elements it finds in the XML config files.

Response:

```
<resourceItemInfo>
    ... standard resourceItemInfo elements ...
</resourceItemInfo>
```

resourceItemInfo contains the same elements as elsewhere.

17.3.1 Upload and Download Locations

After the book is encrypted, the Content Server 4 packaging service needs to write the resulted encrypted file to a location on the media server (upload location) and to record its URL in the database (download location). Upload location can be either a file name (including UNC file name) or an FTP URL (including username/password). Download location must be an HTTP or HTTPS URL. Upload location must correspond to the download location, e.g. when a file is uploaded to the upload location it should become available under the URL specified in the download location. In general, it is not possible to derive the download location from the upload location, so both must be specified.

Examples:

1. HTTP server is running on the Windows server named mymedia (fully-qualified name mymedia.example.com) and a root of HTTP server is shared as "public" through Windows file sharing. You have created a folder "books" where you plan to store packaged books. Then the upload location for a book named Publication.epub be \\mymedia\public\books\Publication.epub and the download location will be *http://mymedia.example.com/books/Publication.epub*
2. HTTP server is running on a remote server named allmedia (fully-qualified name allmedia.example.com). You only have access to a folder which is exposed as *http://allmedia.example.com/library/ebooks*. To upload resources there you have to FTP to ftpmedia server with the user name "books" and password "biblio3" and then go to the folder public. In addition, you have created a folder named math101 where you want to place a book named Calculus1.pdf. The upload location will be *ftp://books:biblio3@ftpmedia/public/math101/Calculus1.pdf* and the download location will be *http://allmedia.example.com/library/books/math101/Calculus1.pdf*.
3. HTTP server is run by a third-party provider and they give you a folder named *http://bigmedia.example.com/bookstore/media*. To upload the books there you have to place them in a local folder C:\upload (on the machine where the packaging service is running) and run a script manually. Then for a book named book.epub, the upload location is C:\upload\book.epub and the download location is *http://bigmedia.example.com/bookstore/media/book.epub*. Note that you still have to run an upload script once the packaging is complete and you can only make the book available to distributors one it becomes downloadable from the download location.

Upload and download locations should be provided in the packaging request using src element (for download location) and location element (for upload location). Another approach would be to configure the packaging service with the base upload location (com.adobe.adept.packaging.baseLocation property) and base download location (com.adobe.adept.packaging.baseURL property). Packaging service will build actual upload and download locations using the base locations and the file name (supplied by fileName element in the request, or resource id with .epub/.pdf suffix if fileName was not specified). Base locations are also used to upload the thumbnail (if it is provided).